

For superior OS-9 performance, the
SYSTEM V

Provides a 68020 running at 25 MHz, up to 128 MBytes of 0 wait-state memory, SCSI and IDE interfaces, 4 serial and 2 parallel ports, 5 16-bit and 2 8-bit ISA slots and much more. The SYSTEM V builds on the design concepts proven in the SYSTEM IV providing maximum flexibility and inexpensive expandability.

AN OS-9 FIRST - the MICROPROCESSOR is mounted on a daughter board which plugs onto the motherboard. This will permit low cost upgrades in the future when even greater performance is required.

G-WINDOWS benchmark performance index for the SYSTEM V using a standard PC VGA board is 0.15 seconds faster than a 68030 running at 30 MHz with ACRTC video board (85.90 seconds vs 86.05 seconds).

Or, for less demanding requirements, the
SYSTEM IV

The perfect, low cost, high-quality and high-performance OS-9 computer serving customers world-wide. Designed for and accepted by industry. Ideal low-cost work-station, development platform or just plain fun machine. Powerful, flexible and expandable inexpensively. Uses a 68000 microprocessor running at 16 MHz.

G-WINDOWS
NOW AVAILABLE FOR
OS-9000

More vendors are of G-WINDOWS with their hardware and more users are demanding it.

A PROVEN WINNER!

Available for the SYSTEM IV and SYSTEM V computers, the PT68K4 board from Peripheral Technology, the CD68X20 board Computer Design Services and now, for computers running OS-9000 using 386/486 microprocessors.

OS-9/68000 SOFTWARE

CONTROLCALC - Real-Time Spread Sheet
SCULPTOR - Development and Run-Time Systems
DATADEX - Free Form Data Management Program
VED ENCHANCED - Text Editor
VPRINT - Print Formatter
QUICK ED - Screen Editor and Text Formatter
M6809 - OS-9 6809 Emulator/Interpreter
FLEXELINT - C Source Code Checker
IMP - Intelligent MAKE Program
DISASM_09 - OS9 Disassembler
PROFILE - Program Profiler
WINDOWS - C Source Code Windowing Library
UTILITIES

Distributor of Microware Systems
Corporation Software

delmar co

P.O. Box 78 - 5238 Surratt Bridge Road • Middletown, DE 19709
302-378-2555 FAX 302-378-2556

Volume One, Issue Eleven

\$3.00

The "International" OS9 Underground®

A Fat Cat® Publication

Magazine Dedicated to OS-9/OSK Users Everywhere!



New Lower Prices! from ColorSystems

Variations of Solitaire

Includes FIVE Variations, Pyramid, Klondike, Spider, Poker and Canfield. Complete documentation shows how to create your own games boot disk using special menu program which is included.

CoCo3 Version \$29.95

MM/1 Version \$39.95

WPSHel

A Word Processing Oriented Point and Click Shell for all your word processing needs. Requires WindInt from your Multi-Vue Disk. Does not include editor, Formatter or Spelling Checker.

CoCo3 Only! \$20.00

We accept Personal Checks or Money Orders drawn from US Banks or International Postal Orders. NC residents please add 6% Sales Tax. Call or write for a FREE catalog!

Please add \$3 per item for shipping outside of the Continental United States.

Quality OS-9 Software for the Color Computer 3 and MM/1 from IMS

NEW!

Using AWK With OS-9

A description of the AWK Programming language with an emphasis on GNU AWK for OSK. Includes the latest version of GNU AWK.

OSK Only! Just \$19.95

OS-9 Game Pack

Includes FIVE complete games, Othello, Yahtzee, Minefield, KnightsBridge and Battleship. Includes special menu program and step by step instructions on creating your own games boot disk.

CoCo3 Version \$29.95

MM/1 Version \$39.95

All CoCo3 Programs require at least 256K of memory

Coming SOON! Indexed Files for OS-9 Level 2, OS-9/68000 and OS-9000!

ColorSystems

P.O. Box 540

Castle Hayne, NC 28429

(919) 675-1706

FAT CAT PUBLICATIONS® BRINGS BACK THE PAST...

HELP SUPPORT THE OS-9 USERS GROUP... AND GET A SPECIAL 3 ISSUE SET OF THE MOTD!

These MOTD's are issues from the "Boisy-Era", edited by Alan Sheltra, editor of the OS9 Underground (and former MOTD Editor). All 3 issues come bound together in a plastic cover and are 8-1/2" x 11". This is Only available for a limited time.

ALL Profits from this Special Re-Print will go to the newly reformed OS-9 Users Group (currently under the trusteeship of Carl Boll). So here's a way to help yourself and the User Group!

SPECIAL PRICE... ONLY \$7.50!

Special
Edition

Set of The OSKer (pronounced Oscar) Magazine

This Special Edition Set of the OSKer Magazine, News and Views in the World of OS-9/68000 and 6809, is the complete set that were published (Issues 1 thru 6). That's over 140 pages of OS-9 reading enjoyment! All pages are "comb-bound", so pages lay flat. Pages are printed to full 8 1/2" x 11" size.

Fat Cat Publications has made special arrangement with the original publisher for this reprint.

Introductory Price of \$15⁰⁰ (till Dec 15th 1993)

(after Dec 15th, 1993 price is \$19.95)

Coming Soon...

The OS9 Underground "Shell Game" 1994 Calendar

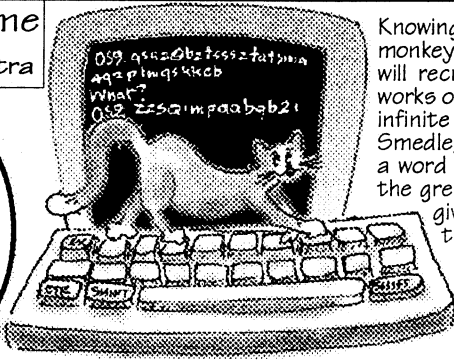
(Shipping is included in all above prices)

Send a check or M.O. to:
Fat Cat Publications
4650 Cahuenga Blvd., Ste #7
Toluca Lake, CA 91602



**FAT CAT
Publications
Bookshelf**

Shell Game
by Alan Sheltra



Knowing that a room full of monkeys with typewriters will recreate all the great works of literature given an infinite amount of time, Smedley figures one cat on a word processor will write the greatest C program given a little more time.

AS'93

Coming Soon... "Shell Game" OS9 Underground 1994 Calendar

Advertisers Index

Vendor	Page
ColorSystems	IFC
CoNect	5
BlackHawk Enterprises	7
How to reach Fat Cat Publications	9
Bob van der Poel Software	10
JWT Enterprises	14
ARK Systems USA	17
OS9 Underground	19
Peripheral Technologies	19
Northern Xposure	21
Sub-Etha Software	23
Farna Systems	24
Dirt Cheap Computer Stuff Co.	29
09-Online	33
OS9 Underground	37
Fat Cat Publication	IBC
DELMAR Co.	BC

The "International" OS9 Underground Magazine

Dedicated to OS-9/OSK Users Everywhere

Volume 1, Issue 11
CONTENTS



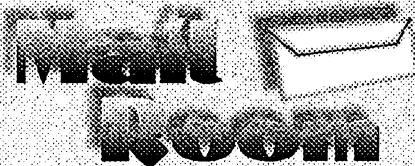
The Underground Staff

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

Mall Room (Letters to the Editor)	4
Under It All (Editor's Column)	6
"Dual'n RS232's" (CoCo Hardware) by Richard Albers	8
"Computer Science 201" "Lesson 3: Doubly Linked Lists" by Scott McGee	16
"Where Am I?" "PWD in B09" by Wayne Campbell	20
"BASIC Training" "Features BPWD by Mike Guzzi" by Jim Vestal	26
"Tales from the Crypt" "Data Encryption in C" by Bob van der Poel	28
"Writing a Device Driver - Part 4" by Boisy G. Pitre	31
"Listing in ASM - SEList" by Allen C.Huffman	34
OS9 Underground Member Card Vendor list	37
Shell Game (Cartoon)	38
Advertiser's Index	38

Fat Cat Publications and The "International" OS9 Underground Magazine and its logotypes are registered trademarks. Subscription rates are \$18.00 for 12 issues (\$23.00 Canada, \$27.00 overseas US Funds). Single or back-issues are available at the cover price (please call or write for availability). Fat Cat Publications is located at 4650 Cahuenga Blvd., Ste #7, Toluca Lake, CA 91602 • (818) 761-4135 (Voice), (818) 365-0477 (Fax) or (818) 769-1938 (Modem). The contents of these pages are copyrighted. Photocopies or illegal reproduction of this magazine in part or whole is strictly prohibited without prior written permission.

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



Any "DiskMaster's" out there?

Enjoyed "OS9 Underground" mensely; keep up the good work.

Does anyone out there have one of the D.P. Johnson's DiskMasters? God I hope I am not the only one who bought the thing. I would like to contact [other] users. Also, do you have any info on the Level 2 versions. I have V2.01. I see reference to V2.4. Am I missing something?

-Leonard H. Reed, Sr., Gold Hill, OR.

I know of 2 DiskMaster user's, one in Canada, (Dieter Rossmann), and one in your own home state, (Paul Pollock). I'm sure there are others out there as well.

The version V2.4 you are referring to is for OS-9/68K, (OSK). The last, (released), version for the CoCo is the one you have.

-Editor

Feedback on Proposed Magazine Changes...

[In response to the reader survey card]

That special binding you're considering will likely cause the envelope to tear. Why not consider 2 color printing and "real" type-setting and offset printing?

-Bruce Moore, Chantilly, VA

Bruce, since you are in the printing business you might be familiar with "comb-binding". This binding process is very lightweight and will not tear the envelope no matter how rough the Post Office handles it. I have used this method before and never had a document, bound in this fashion, tear an envelope.

While I would love to send my output to a linotronic typesetter (1200 to 2400 DPI), and add two color printing, the cost would be extremely prohibitive. Unfortunately, the subscriber base does not yet make those "niceties" cost effective as yet. Someday though!

-Editor

Ooops...

From: PHXKEN@delphi.com
 Subj: autofmt_OS9Underground
 To: ZOGster (On Delphi)

Alan, Paul Fitch has helped me on FIDO to make the basic09 program on page 16 work correctly. Several spaces were not shown on the SHELL call line of autofmt. There should spaces before and after the "r" and there seems to be a space missing after the word Format. The line should read:

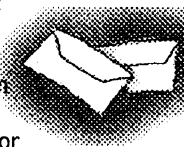
SHELL "Format "+drivename+" r "+
 CHR\$(34)+diskname+CHR\$(34)

-Farrell Kenimer alias Ken/Az

Thank you for pointing out that error, Ken. This was an error, most likely, created by me rather than the author though.

-Editor

Letters to the Editor may be sent to:
 4650 Cahuenga Blvd., Ste #7
 Toluca Lake, CA 91602
 or as email to the
 following addresses:
 EditorOS9U@AOL.com
 -or -
 OS9Under@AOL.com or
 to "TheFatCat" Mail List:
 "TheFatCat@AOL.com" (See Page 9)
 "ZOGster" (on Delphi)



Get Noticed... Advertise in The OS9 Underground...

Our ad rates are something you can cheer about!

Call or write for our Ad Rate Card
 (818) 761-4135 • (818) 365-0477 (fax)
 4650 Cahuenga Blvd., Ste #7
 Toluca Lake, CA 91602

OS9 Underground Magazine Member Card

Participating Vendors

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

- CoNect 10% Off any order
- Sub-Etha 10% Off any order
- Canaware 10% Off WristSavers
 10% Off WristSaver MousePad & 15% Off ENC9
- AniMajik Productions 20% Off all Software
- Farna Systems 10% Off any order

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

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

Coming Next Month... **Holiday Issue**

LINEFEED STX POINTER
 LDA #StdOut
 LEAX LF,PCR
 LDY #2
 OS9 ISWrite output carriage return/line feed
 LDX POINTER
 LBRA EXEC go back to start...
 * Display help message.
 HELP LDA #StdOut 1=std out
 LEAX HELPMMSG,PCR point X to help message
 LDY #helpsize load Y with length of help message
 OS9 ISWrite write help message to screen
 BRA EXIT goto EXIT
 * Get prompt text.
 GETPROMPT
 CLR COUNT,U clear COUNTER
 STX PMTPTR store location of prompt text in
 PMTPTR
 GLOOP LDA ,X+ X points to command line, load
 character in A
 CMPA #" is it a quote?
 BEQ GEND if yes, goto GEND
 CMPA #SD is it enter?
 BEQ GEND if yes, goto GEND
 INC COUNT,U increment COUNTER
 BRA GLOOP goto GLOOP
 GEND LDA COUNT load COUNT into A
 INC COUNT,U increment it (0-9 becomes 1-10,
 basically...)
 STA PMTLEN store length of prompt in PMTLEN
 LBRA EXEC goto EXEC
 * Return to OS9.
 EXIT CLRB
 ERROR STB COUNT store error code in temporary
 COUNTER
 LEAX BUFFER,U point X to tmode buffer
 LDA ECHO load A with original echo status
 STA +4,X restore it
 LDA PAUSE load A with original pause status
 STA +7,X restore it
 LDA #StdIn 0=std in (current path)
 CLRB 0=ss.opt
 OS9 ISSetStt restore tmode settings
 LDB COUNT load B with saved error code
 OS9 F\$Exit then exit
 PROMPT
 BSR PROMPTER output prompt
 LDX POINTER point X to command line position
 LDA -1,X decrement (go back on character)
 CMPA #SD is it ENTER?
 LBEQ EXIT if so, done...

LBRA EXEC else go back to start
 * Display prompt, wait for keypress, eradicate prompt,
 format hard drive.
 PROMPTER
 LDB PMTLEN load B with how many characters
 to output
 STB COUNT store it in COUNTER (to be used in
 a moment)
 CLRA A=0 (building D register from A/B, D=rArB)
 TFR D,Y transfer D to Y register
 LDA #StdOut 1=std out
 LDX PMTPTR point X to prompt text
 OS9 ISWrite write prompt to screen
 BCS ERROR if error, goto ERROR
 * Read one character.
 CLRA 0=std in
 LEAX LINEBUF,U point X to line buffer
 LDY #1 load Y with max. # of characters to read
 OS9 ISRead read character
 BCS ERROR if error, goto ERROR
 * Backspace over whatever prompt was used.
 LDA #StdOut std out
 LEAX BACKSPACE,PCR point X to backspace text
 LDY #3 load Y with max. # of characters to output
 LOOP OS9 ISWrite write backspace text to screen
 BCS ERROR if error, goto ERROR
 DEC COUNT decrement COUNTER
 BNE LOOP if COUNTER not 0, goto LOOP
 RTS
 PROMPTLEN FCB 22
 PROMPTMSG FCC "Press [Enter/Return] :....."
 BACKSPACE FCB \$08,\$20,\$08
 FILEMESS FCC "[Press (A)abort, (Q)uit, or (S)top to
 Abort File]"
 FCB \$D,\$A
 LF FCB \$D,\$A
 filesize equ *-FILEMESS
 HELPMMSG FCB \$D,\$A
 FCC "SELister 1.05 - By Allen Huffman of Sub-Etha Software."
 FCB \$D,\$A
 FCC "Syntax: SELister 'Prompt' filename! {prompt?} (filename?)
 ..."
 FCB \$D,\$A
 FCC "Usage : Displays text file(s) with page pausing
 prompt, which is"
 FCB \$D,\$A
 FCC" changeable. Also functions as a stand-alone
 'pause' command."
 FCB \$D,\$A
 FCC " Multiple files and/or prompts may be used."
 FCB \$D,\$A
 helpsize equ *-HELPMMSG
 ENDSECT



CoNect New Hardware

Mini-RS232 Port: If you are in the market for a Tandy-compatible serial port, check out our Mini! This ROMPak unit supports all seven control lines available, and can supply more output current than even the Tandy Pak! Jumper selectable addresses and cd swap.

Only \$49.95

(Y Cable use requires 12 volt power supply, add \$9.95)

XPander: The XPander allows you to assemble the most compact CoCo3 system possible using a stock motherboard. For example, the electronics of my 2 meg CoCo3 with Tandy Floppy Controller, Burke & BurkeXT, WD1002 Hard Drive controller, rs232 port, puppo and Hi-Res adaptors from a block 12 inches long, 7 inches deep and 3 inches at it highest point.. Not only will this fit in the smaller PC cases,, but in a modified CoCo case.

Obviously this is not a full tilt MultiPak clone - there just isn't room. The two external slots may both contain /scs decoded devices, but only one slot ROM may be used. The external slot may be used either as a ROMPak port (disables internal hardware when Pak is inserted), or as an undecoded buss slot. 12v is available at all slots.

The no-slot RS232 port is a virtual clone of the mini-rs232 described above, and saves not only a slot but quite a bit of room in the finished package.

The Xpander is available in two versions. If a PC type case/power supply will be used, order just the board. CoCo Kit includes a new lower case shell and 450ma +/- 12v power supply.

CoNect Custom CoCo Cables!

Cassette - (mini or rca)	\$ 4.95	Comp. Video (6 feet)	\$ 3.95
Disk Power (either way)	\$ 5.95	Printer (DIN to DIN)	\$ 4.95
2 Drive Floppy Data	\$14.95	3 Drive Floppy Data	\$19.95
RS232 (db 25, db9, din4 to db25,db9 - normal, nulmodem, or dcd swap)			\$ 9.95

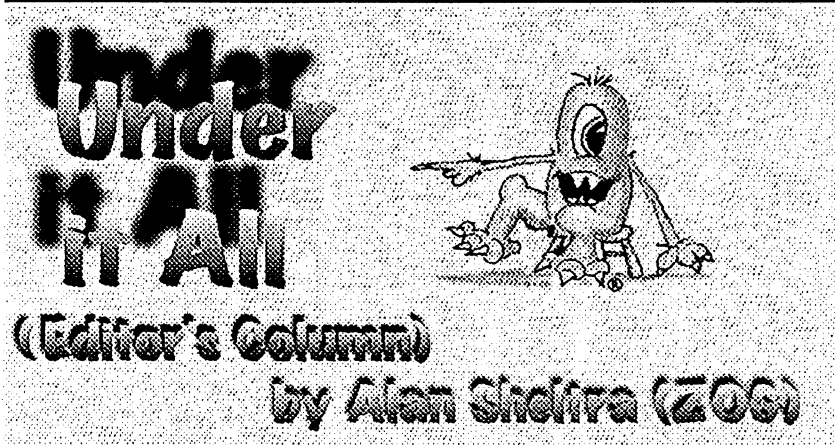
Ram Upgrades

64K CoCo 1 (F board), or CoCo2 (2 or 8 chip) with instructions	\$ 7.95
512K CoCo3 (various makes) not always available	\$49.95
2 Meg CoCo3 (Disto) with SIMMS	\$194.95
Installed (add 11.95 for 6309)	\$219.95

CoNect

449 South 90th Street
Milwaukee, WI 53214

(414) 258-2989 • Delphi: RICKULAND • Internet: rickuland@delphi.com



Rocket Crash

Our lead story, in issue 8, as some of you may recall, was "The Rocket, but will it Fly?". Well, it seems it's destined not to. Here's an excerpt of a message reply from Chris Burke on Internet:

[Zack Sessions notes:]

>As I understand it, one of the main reasons Chris >decided to not produce the Rocket was that >Microware screwed him on the OSK license quote.

Ouch! As noted in a previous posting, there are no villains here. Microware changed their product line, and with it their pricing. The new pricing made it significantly less attractive for Burke & Burke to enter the OSK hardware market with a product which would ship only 100 units. There are some real advantages to the new pricing, but in our case it essentially doubled our production costs. Depending on our ship date, we might have been able to get the older Microware pricing and product. Rocket orders never reached the 100 unit mark, so Burke & Burke couldn't commit to an order. Microware was both flexible and generous, but the deal just didn't work out.

Chris made an honest effort with the "Rocket" board for the CoCo and it's really a shame things did not work out as planned. It would have given the CoCo a much needed kick in the pants.

DELMAR Announces Port of GWindows to the MM/1

Ed Gresick, of DELMAR CO., asks: "Are you interested in G-WINDOWS for the MM/1? If so, I am willing to do a port of G-WINDOWS to the MM/1 Computer. Price will be \$200.00 per copy." "This will include the software, all the G-WINDOWS utilities, demo programs, images, and sample source code. Documentation includes the G-WINDOWS manual and a tutorial prepared by DELMAR CO." "Before starting the port, I will require twenty (20) confirmed orders. At this point, I have received 8 confirmed orders. After I receive the necessary commitments, 2 to 3 months will be required to do the port. Orders may be made by mail or e-mail accompanied by check or credit card information. Checks and credit card payments will be held until 20 orders are received or December 31, 1993. If the necessary orders are not received by December 31, 1993, checks will be returned and credit card payments not processed.

Call Ed at (302) 378-2555 for more info. If you have an MM/1.. go for it! I have been using GWindows on my TC70 for over a year now and really enjoy it.

Outta Space.. Again!

Returning next month is Eric Levinson's column as well as our BBS Listing we had hoped to fit in this issue. Also, Allen Huffman's COMPLETE Atlanta Fest Report! C'ya next month!
-Alan Sheltra (ZOG)

STA PMTLEN,U	store it in PMTLEN	OS9 ISReadLn	read line
LEAX PROMPTMSG,PCR	point X to default prompt	BCS LERROR	if error, goto LERROR
STX PMTPTR,U	store it in PMTPTR	LDA #StdOut	1=std out
LDA #StdIn	0=std in	OS9 ISWritLn	write line to std out
CLRB	0=ss.opt function	LBCS ERROR	if error, goto ERROR
LEAX BUFFER,U	point X to buffer	LDA #StdIn	standard input
OS9 ISGetSt	read tmode settings into BUFFER	LDB #1	_gs_rdy (character ready syscall)
LBCS ERROR	if error, goto ERROR	OS9 ISGetSt	check to see if character ready...
LDA +4,X	load A with current echo status	BCS LCONT	if nothing there, continue
STA ECHO	save it for later in ECHO	LEAX LINEBUF,U	point X to line buffer
LDA +7,X	load A with current pause status	LDY #1	1 character to read
STA PAUSE	save it for later, also, in PAUSE	OS9 ISRead	read it
CLRA	set A to 0	LDA LINEBUF	load A with character gotten at prompt
STA +4,X	turn off echo	ANDA #SDF	make upper case
STA +7,X	...and pause	CMPA #A	is it Abort?
LDA +8,X	load A with lines per page	BEQ LABORT	
DECA	decrement A (make room for PAUSE msg on screen)	CMPA #Q	is it Quit?
STA PAGELEN	store it in PAGELEN	BEQ LABORT	
LDA #StdIn	0=std in (current path) {again}	CMPA #S	is it Stop?
CLRB	0=ss.opt function {also again}	BEQ LABORT	
OS9 ISSetSt	update status	LCONT	
LDX POINTER	restore X from POINTER	DEC COUNT	decrement COUNTER
* Scan command line for options/parameters.		BNE LIST	if not 0, continue at LIST
EXEC LDA ,X+	X points to command line, load char in A	LBSR PROMPTER	display prompt
CMPA #32	is it a space?	LDA LINEBUF	load A with character gotten at prompt
BEQ EXEC	if yes, goto EXEC and keep scanning	ANDA #SDF	make upper case
STX POINTER	save X in POINTER	CMPA #A	is it Abort?
CMPA #SD	is it enter?	BEQ LABORT	
LBEQ PROMPT	if yes, goto PROMPT	CMPA #Q	is it Quit?
CMPA #-	is it a dash?	BEQ LABORT	
LBEQ HELP	if yes, goto HELP	CMPA #S	is it Stop?
CMPA #"	is it a quote?	BEQ LABORT	
LBEQ GETPROMPT	if yes, goto GETPROMPT	LDA PAGELEN	load A with page length
* If here, assume it must be a filename.		STA COUNT	store it in COUNTER
LEAX -1,X	decrement X (BRUTE FORCE!!!)	BRA LIST	goto LIST
LDA #1	1=read acces	LABORT LDA PATH	get filepath
OS9 ISOpen	open filename pointed to by X	OS9 ISClose	close it
LBCS ERROR	if error, goto ERROR	BCS ERROR	
STA PATH	store path in PATH	LDX POINTER	restore command line position
STX POINTER	store new command line ptr	LDA ,X	get character
* position in POINTER		CMPA #SD	are we at end of line?
LDA PAGELEN	load A with page length	LBNE EXEC	no, so continue scanning line
STA COUNT	store it in COUNTER	BRA EXIT	otherwise done.
LDA #StdOut	Display abort message...	LERROR CMPB #EEOF	is error code "end of file"?
LEAX FILEMESS,PCR	point to file message	BNE EXIT	if no, goto EXIT
LDY #filesize	length of message	LDA PATH	load path into A
OS9 ISWrite	output it	OS9 ISClose	close the file
DEC COUNT,U	decrement count twice to compensate for	BCS ERROR	if error, goto ERROR
DEC COUNT,U	two lines of text already printed	LDX POINTER	this is some redundant code, but I think I must
LIST LDA PATH	load path into A	LDA ,X	do it this way. IE, (BRUTE FORCE!!!)
LEAX LINEBUF,U	point X to line buffer	CMPA #SD	
LDY #80	load Y with max. # of chars to read	BNE PROMPT	

LISTing in ASM

by Allen C. Huffman

When I was running an OS-9 BBS system, I decided I needed a more powerful way to display text files to my users than the normal "LIST" command. SELister is the result. SELister is a replacement for the normal "LIST" command but with some interesting modifications: It pauses the listing file at the end of each screen, it allows the user to abort the file while it is listing, and it gives a settable prompt to use either by itself or as the "end of screen" prompt when listing files. It will also handle multiple files in the sequence that you enter them on the command line. Here are some examples:

```
selister filename.txt
```

The above lists the file, pausing at end of each screen with a default prompt.

```
selister "[PAUSED]" filename.txt
```

This lists the file, pausing at the end of each screen using message "[PAUSED]".

```
selister "<Hit a Key>" filename.txt "<Press ENTER>"
```

This lists the file using "<Hit a Key>" as the screen pause prompt, then prompts with "<Press ENTER>" after the file has been displayed.

```
selister
```

By itself it displays a default "Press [Enter/Return] :" message.

```
selister filename1.txt filename2.txt  
"<PAUSE>" filename3.txt
```

shows the first two files using the default prompt, then shows the last file

using the "<PAUSE>" prompt.

Everything else should be fairly self-explanatory. "selister -?" will display a help message. RMA source code is provided, as well as a BASIC09 program that will create the executable file. Extra space was left after the default prompt text to allow patching a different default prompt (up to 32 characters). If this is done, the first byte before the prompt text must be changed to the number of characters in the prompt. Enjoy!

Note: This routine is Copyright (C) 1993 by Allen Huffman of Sub-Etha Software. Use it as you wish, but please do not distribute modified versions without the author's consent.

-Allen C.Huffman

```
* SELister 1.05 by Allen C. Huffman
* Copyright (C) 1993 by Sub-Etha Software
* Single/multiple file lister/prompter utility thing.
* 0.00 08/26/92 - initial version
* 0.01 06/27/92
* 0.02 08/31/92
* 0.03 09/10/92
* 0.04 10/22/92
* 1.04 11/21/92 - scan for "abort" after each line
* 1.05 07/17/93 - minor code change, update usage message
```

```
StdIn equ 0
StdOut equ 1
StdErr equ 2
PSECT SELister,$11,$81,0,200,START
```

```
VSECT
PATH RMB 1
COUNT RMB 1
PAGELEN RMB 1
ECHO RMB 1
PAUSE RMB 1
PMTLEN RMB 1
PMTPTR RMB 2
POINTER RMB 2
LINEBUF RMB 80
BUFFER RMB 32
ENDSECT
```

* Setup. Point to default prompt. Turn off echo and pause, etc...

```
START
STX POINTER save X in POINTER
LDA PROMPTLEN,PCR load A with length of prompt
```

BlackHawk Enterprises

P.O. Box 10552
Enid, OK 73706-0552
(405) 234-2347

Announcing!

The 68340 Accelerator Card for the MM/1!

Produced by Kreider Electronics, and marketed by BlackHawk Enterprises and representatives of IMS, the Accelerator Card brings 68020 power to users of the MM/1! The 68340 mpu features :

- A 68020 core minus bitfield instructions
- 3 serial ports with hardware handshaking vs. 2 serial ports, one without handshaking on the 68070 board.
- No more 64K limit on DMA access.
- Eprom boots from the hard drive,

MORE SPEED!

Programs compiled on an accelerated MM/1 (MM/1a) with ULTRA C (not included), benchmarked at 5770 dhrystones. An MM/1a with MWC 3.2 yielded 2777 dhrystones, while an unmodified MM/1 yielded 1000 dhrystones from the same code. So, the 68340 upgrade gives you a speed increase ranging from 2.7 to 5.7 times as fast as that of a stock MM/1!

Call today, and order your 68340.

At \$325, this is a real bargain!

From HyperTech Software!

DeskTop 1.0	\$79	Paint 1.0	\$79
PixUtils	\$24	Fontasee	\$34

Bob van der Poel Software!

VED 2.0	\$59.95	VPRINT	\$59.95
---------	---------	--------	---------

Call for other products today!

Dual'n RS232 Ports

by Richard Albers

As faster modems have become more affordable, they have become more and more popular. Fast modems are replaced with even faster ones, and the prices of the "old, slow" ones can be expected to drop. "Old" and "slow" are, of course, relative terms. The first serial circuits I worked on were "new and fast" 60 baud! Yes, 60; up from 50. A very few years ago 9600 was state of the art. Soon mega-bauds will abound. CoCo may not go quite that fast, but it can do 9600 or more. If you want to upgrade a Tandy Direct Connect Modem Pak (DCM) to more than 300 baud, and maybe with two ports, this is for you. I will describe how to remove the parts that make up the DCM's 300 baud-only modem and substitute new parts to interface your CoCo to an external 300 to 19200 baud modem, and optionally, add a second serial port with the same capabilities. You can build one port now and add a second port later. If you can solder small parts, use common hand tools, and follow instructions, you can convert a DCM pak to work like the popular Deluxe RS232 pak. While this is not a job for a beginner hacker, it makes a good intermediate level project.

Read the instructions several times until you understand all the steps. If you feel this project is too advanced for you, perhaps you can find someone to do the parts you don't want to do, in exchange

for doing parts of their conversion. This could even be a club construction project.

You will need a few basic tools: Screwdrivers - a #1 Phillips and a small fl blade, small needle-nose pliers Diagonal cutter (flush-cutting is nice), large n clipper (straight cutting edges are best), a sharp knife, such as a Stanley utility knife, a small file and/or sandpaper A wire wrapper/unwrapper/stripper, a 25 watt and a 40 watt or controlled temperature soldering iron, small diameter solder (about 1/32 inch diameter), vacuum device or braid for removing solder, a small drill with 5/32" and 1/16" or smaller bits, a DVM (best), VOM, or continuity tester. These are the minimum; no hacker ever has too many tools!

Some new parts - for dual ports:

2 - AT type serial cables
(with DB9S and DB25P connectors)
2 - Right angle PCB mount DB9P connectors to fit cables
1 6551A ACIA and
2 - 1488 and 2 - 1489 ICs
1 - 1.8432 MHz crystal similar to X1 in the DCM.

For one port:

1 - AT type serial cable (with DB9S and DB25P connectors)
1 - Right angle PCB mount DB9P connector to fit cable
1 - 1488 IC
1 - 1489 IC

Note: The PCB mount DB9P connectors must have exposed conductors between the connector end and the PCB end. That's where the wires will connect. Get cables with the narrowest ends you can find. They should be no wider than the connectors they will plug into on the 9 pin end. You can substitute RJ45 (8 pin telephone type) connectors for the DB types. Get a cable first, then wire the RJ45 to put the signals on the correct pins of the

```
* move.b VOut1(a3),(a0)+ move byte into (a0) and increment
move.w d2,d1 move computed driver delay into d1
RecDelay subi.w #1,d1 subtract 1 from delay value
bne.s RecDelay
subi.l #2,RSd2(a5)
bhs.s RecLoop subtract 1 from buffer size
clr.w d1 return SUCCESS
rts return to caller
```

```
Unknown: move.w #ESUnkSvc,d1 Unknown service code
ori.b #Carry,ccr return Carry set
GetSta99: rts
<End of Figure 1>
```

Figure 2

```
*****
* SetStat: set device status
*
* Passed: (a1) = Path Descriptor
*         (a2) = Static Storage address
*         (a4) = process descriptor
*         (a5) = caller's register stack ptr
*         (a6) = system global data ptr
*         d0.w = status call function code
*
* Returns: varies with function code
*
* Error Return: (cc) = carry set
*              d1.w = error code
*
* SS_Play parameters:
*
*   a0.l = Pointer to buffer
*   d0.w = Path number (unused)
*   d1.w = Function code
*   d2.l = Buffer size
*   d3.w = Sample rate
```

```
SetStat:
cmpi.w #SS_Play,d0 is it the play option?
bne.s SetStat10 no. check next possibility
move.l RSd3(a5),d2 move sample rate into d2.l
divu.w #LoopConst,d2 divide by loop constant
and.l #SFFFF,d2 throw away upper 16 bits
movea RSa0(a5),a0 move buffer ptr into a0
movea V_PORT(a2),a3 move base A/D address into a3
PlayLoop move.b (a0)+,VOut0(a3) move byte into
channel 0
move.b (a0)+,VOut1(a3) move byte into channel 1
move.w d2,d1 move computed driver delay into d1
PlayDelay subi.w #1,d1 subtract 1 from d1
bne.s PlayDelay loop if not zero
subi.l #2,RSd2(a5) subtract 2 from buffer size
bpl.s PlayLoop branch if buffer > -1
SetStatOK clr.w d1 return SUCCESS
PutSta99: rts
```

```
* Dummy functions to appease the kernel
SetStat10 cmpi.w #SS_Open,d0 check for Open
beq.s SetStatOK
cmpi.w #SS_Release,d0 check for Release
beq.s SetStatOK
cmpi.w #SS_Close,d0 check for Close
beq.s SetStatOK
bra.s Unknown
```

<End of Figure 2>

-Boisy G. Pitre



09-Online Systems

New Product

Technology Concepts
LineLink 14.4 v.42bis
Data/Fax External Modem

Only \$125.00!

(Price Includes shipping)

Send check/M.O. made payable
to: J. E. Vestal

09-Online Systems

c/o Jim Vestal
221 E. 17th, #31
Marysville, CA 95901

Ask to be placed on our mailing list
for a Free shareware catalog.

into play (such as caching) and even the type of processor you're running. For my 20MHz 68030, I've chosen a delay of 110.

THE GETSTAT ROUTINE

Figure 1 shows the source code for the GetStat call. The SS_Record parameters attempt to follow the standard GetStat routine by providing for d0.w as the path number (although it isn't even referenced by the routine). a0.l holds the pointer to a pre-allocated buffer, d1.w has the function code SS_Record, d2.l holds the size of the buffer who's pointer was passed in a0, and d3.w holds the sampling delay in hertz (e.g 22KHz = 22000 in d3.w)

Upon entering the routine, we compare the SS.Record value with the value in register d0.w (which holds the status call function code). If another function code was passed, a branch to Unknown: is made, and the routine exits with an "Unknown Service Request" error code.

If the function code is SS_Record, we move the sample rate into d2. and divide that value by the loop constant. The resulting value is then ANDed with \$FFFF to throw away the upper 16 bits of the long word. Now d2.l contains the loop delay which will be used by the driver when recording sound.

The caller's address buffer is then moved into a3 and bytes are subsequently moved from the A/D port into the buffer while a0 is incremented to accommodate the next byte. We then move the driver delay we computed earlier into d1, and subtract 1 from that value until we reach zero.

We then subtract 1 from the buffer size and loop back to reading from the A/D port until the size reaches zero (meaning we have reached the end of the buffer).

THE SETSTAT ROUTINE

The SetStat routine is similar to the GetStat except that a pointer to valid data is handed to the driver routine and is

passed to the A/D port.

Notice that this routine also contains "dummy functions" which are called at various intervals by the kernel. SS_Open, SS_Relea, and SS_Close are called by the kernel to open, release and close a device. This used by drivers to manipulate baud rate and parity parameters. Since our driver doesn't use them, we just clear d1 and return normally.

IN CONCLUSION

Figure 3* is the driver in its entirety. While there are many issues that remain open, (such as interrupt service routines and minimal GetStat/SetStat calls), I hope you've gained some insight on the basics of OS-9 driver writing.

*Editor's Note: Due to space Figure 3 will appear in the Dec. Issue.

```

Figure 1.
*****
* GetStat: get device status
*
* Passed: (a1) = Path Descriptor
*         (a2) = Static Storage address
*         (a4) = process descriptor
*         (a5) = caller's register stack ptr
*         (a6) = system global data ptr
*         d0.w = status call function code
*
* Returns: varies with function code
*
* Error Return: (cc) = carry set
*              d1.w = error code
*
* SS_Record parameters:
*
* a0.l = Pointer to buffer
* d0.w = Path number (unused)
* d1.w = Function code
* d2.l = Buffer size
* d3.w = Sampling delay
*
GetStat:
    cmpi.w #SS_Record,d0 is it the record option?
    bne.s Unknown no, exit w/ error
    move.l R$d3(a5),d2 move sample rate into d2.l
    divu.w #LoopConst,d2 divide by loop constant
    and.l #$FFFF,d2 throw away upper 16 bits
    movea.l R$a0(a5),a0 move caller's a0 (buff ptr) into a0
    movea.l V_PORT(a2),a3 move base A/D address into a0
    RecLoop move.b VOut0(a3),(a0)+ move byte into
    *              (a0) and increment
    
```

DB25 connector to the modem. You can wire one end of the cable to a DB25S (cheaper) or use an RJ45 to DB25 adapter (easier). There are no standards for using RJ45 connectors here, you get to set your own. Just make both cables, (ports), the same. I used glue and cable ties to fasten the RJ45s to the PCB.

30 Gauge wire (several colors preferred) Hardware to mount connectors to PC board Epoxy or instant glue (crazy glue), and of course, a DCM pak to modify and a modem or two.

The parts for a dual port conversion almost total the minimum order of \$30 from Jameco (1-800-831-4242). If you shop around, you may reduce the expense by half.

You should already have a CoCo, disk drive and Multi-Pak or equivalent with which to use your modified DCM pak.

Some warnings: No one guarantees your success. The author and staff of this magazine have tried to keep these instructions clear and correct, but the end result depends solely on you the builder.

Shavings and dust should be avoided. This is an old part, and no information is available concerning any hazards involved. Use a vacuum to remove dust and particles so you don't breathe a possible health risk.

You should work in a static free area the ACIA especially is easily damaged by static discharge, and the other parts are not immune. If you get a big spark when you walk across a carpet and touch a grounded object, wait for another day to work on anything electronic!

Test your DCM pak to be sure it works at high speed. This can be troubleshooting time later in case there is a problem with it. It's your choice whether to repair a bad DCM pak, or to just replace it. Very little of the existing circuitry will be used, so any problems

How to reach The OS9 Underground:

For subscription information, questions, call or write to:

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

Article Submissions may be sent to the above addresses or to the following special internet account set up for that purpose:

OS9Under@AOL.com

Submissions should be sent as plain text and unformatted. Join us on the New OS9 Underground Email, Mailing List

TheFatCat@AOL.com

To Subscribe, just send email to the above account and you will be added to the list

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

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



Fat Cat Publications

may be eliminated with the unused parts, but a known good unit will help. This would be a good time to download one or more terminal programs, since we will remove the ROM in the DCM.

Next, test your cables. Cable manufacturers are not perfect, and some cables that look right may be wired strangely. Make sure the correct pins are connected at each end, and that there are no shorts between pins. Repair, replace, or exchange as you prefer. The connections are shown in a table.

Now to the good part: Open the DCM case. Remove the black label on the bottom which will "void your warranty". Remove the screw under it. Release the snaps by pushing the bottom case half toward the switches and the top half toward the edge connector. The connector end will release so you can pull the halves apart. It's easiest to release the snap on one side at a time. If you have opened other packs this will be a familiar operation, if not

just take your time and try to not break the case. Fold the case halves sideways. They will only fold one way because of a short grey cable connecting the main circuit board to a small satellite LED board. Remove one screw and lift the small LED board out. Set the case top aside for now. Remove the 4 screws and lift the PCB (*Printed Circuit Board*) from the case bottom. Fold a strip of aluminum foil over the edge fingers for static protection. Hold it in place with a rubber band. Remember to remove the foil before you plug the board in for testing. You can get accustomed to seeing it there that you don't notice it, and it will short out everything. Use a wire wrapper/unwrapper to push out the 4 pins and remove the shield from the PCB. Set all the parts aside except the PCB.

Now remove the parts you no longer need. The fastest and easiest way is to just cut them off with a diagonal cutter and nail clippers. You may non-destructively re-

Ved/68000 Text Editor 2.0

Our editor just got better! Ved 2.0 now includes an integrated spelling checker! Plus it supports multiple buffers! This is in addition to all the features of the original: user control over macros, key bindings, editing modes; automatic indenting and numbering; wordwrap on or off; search; find/replace; block move/copy/delete; word and line delete; "undo"; and a lot more!

Ved 2.0 also supports your KWindows mouse... but it still works with any terminal (as long as it has cursor positioning).

Ved comes complete with MVEF (an editor for creating the environment files Ved uses for configuration) and a 100 page manual which fits your Microware binders. For more information, just drop us a note and we'll send you full information on Ved and other fine products.

Ved 2.0 complete with the spelling checker and MVEF, costs only \$59.95 plus \$3.00 shipping and handling. To order please send your check or M.O. and preferred disk format to:

Bob van der Poel Software

P.O. Box 355

Porthill, ID

USA 83853

Phone (604) 866-5772

P.O. Box 57

Wynndel, BC

Canada V0B 2N0

CIS: 76510,2203

SS_Record as 150.

TIMING ISSUES

We hit a major stumbling block when dealing with the issue of timing. *Why?* Because most sound files are recorded at ranges from 11KHz to 22KHz. This means that even at a low rate of 11KHz, one byte of data must be read from the A/D port every 1/11000's of a second.

This brings up some problems under a real-time operating system like OS-9.

OS-9's timing granularity, known as a tick, is normally 1/100's of a second. That is 110 times slower than what is necessary to create a sample of mediocre quality. Therefore, we cannot sleep even for one tick while waiting for the next byte to be read — it is simply too long a wait. Nor can we use a cyclic alarm since its granularity is in ticks as well.

Without DMA (as is the case with our example), our only solution is to loop inside the driver. It isn't the most desirable solution, since looping inside a driver keeps OS-9 locked in system state and user processes do not run until the driver's routine is exited. However, interrupts from other devices are still processed (typing on the keyboard while looping in the driver will save the keystrokes in the SCF buffer).

Due to this limitation (which should impress upon you the necessity to use interrupt driven devices), we are forced to compute a delay prior to assembling the driver. This delay value will provide timing base from which our driver will derive the proper delay as close to the real rate as possible. Doing this in the driver prevents applications that use these functions from knowing the details of timing.

The delay value is dependent upon your computer's speed (in cycles per second) as well as the number of cycles it takes to execute the looping code in the driver. Even more variables can come

Writing a Device Driver

Part IV

by Boisy Pitre

This month we finish our analog/digital device driver by adding customized sound record and playback capability.

Recording and playback of sampled sound is a very specific application which doesn't readily fit within the specifications of our driver. However, OS-9 provides an easy means of interfacing specialized operations through the use of GetStat (Get Status) and SetStat (Set Status) calls.

Playback of sound is analogous to "setting" data into our device, while recording data from our device is similar to "getting." It does not matter which routine actually performs the recording or playback of sound; however, consistency's sake we will assign sound playback as a SetStat call and sound recording as a GetStat call.

Now that we have made provisions for recording and playback, we need to define a name and function code for each of the actions. Following Microware's naming convention for status calls, we will define playback as SS_Play, and recording as SS_Record.

We must also assign a unique function code to the respective name.

Codes 0-127 are reserved

Microware's use, leaving us 128-65535.

I've chosen SS_Play as 151 and

```

/* crypt.c */
.....
*   COPYRIGHT NOTICE   *
.....
* This software is
* copyright (C) 1993 by Bob van der Poel Software

* Permission is granted to reproduce and distribute
* this file by any means so long as no fee is charged

* above a nominal handling fee and so long as this
* notice is always included in the copies

* Other rights are reserved except as explicitly granted
* by written permission of the author.

* Bob van der Poel Software
* PO Box 355, Porthill, Id, USA, 83853
* PO Box 57, Wynndel, BC, Canada, V0B 2N0
* Phone 604-866-5772
* CIS 76510,2203
*/

#include <stdio.h>
#include <modes.h>

#define BLOCKSIZE (10*1024) /* use 10K file chunks
for speed */

extern int errno;

char buffer[BLOCKSIZE];

main(argc,argv)
int argc;
char **argv;
{
    register int i, count;
    int inpath,outpath;
    register char *key1ptr, *key2ptr, *buffptr, *keystart,
    *keyend;

    /* no filename, show help */
    if(argc==3) usage();

    /* if any arg starts with -, help */
    for(i=1; i<argc; i++)
    {
        if(argv[i][0]=='-') usage();
    }

    /* open two paths to the file; one read, one write */
    if( (inpath=open(argv[1],S_IREAD))==-1 ||
        (outpath=open(argv[1],S_IWRITE))==-1 )
    {
        fprintf(stderr,"crypt: can't open %s\n",argv[i]);
        exit(errno);
    }

    /* set up pointers to the start/end of the key */
    key1ptr=keystart=argv[2];

```

```

key2ptr=keyend=keystart+strlen(keystart);
if(strlen(keystart)<3)
{
    fprintf(stderr,"crypt: key must be at least 3
characters\n");
    exit(1);
}

/* process the file */
while((i=count=read(inpath,buffer,BLOCKSIZE))>0)
{
    for(buffptr=buffer; --count; )
    {
        if(*key1ptr=='\0')
        {
            key1ptr=keystart;
            key2ptr=keyend;
        }
        *buffptr ^= *key1ptr++;
        *buffptr++ ^= *key2ptr--;
    }
    write(outpath,buffer,i);
}
close(inpath);
close(outpath);

usage()
{
    register int t;

    static char *msgj[]=
    {
        "",
        "Crypt, (c) 1993, Bob van der Poel Software",
        "Usage: crypt <filename> <key>",
        "Function: encrypts a file using a user supplied key",
        "",
        "***** Caution: Don't forget the key—not even god
*****",
        "***** can decrypt the file without the exact key! *****",
        ""
    };

    for(t=0; t<sizeof(msgj)/sizeof(msgj[0]); t++)
    {
        fputs(msgj[t],stderr);
        putc('\n',stderr);
    }

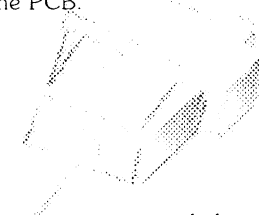
    exit(0);
}

```



move them for a later project if you wish, but remember that the most important part for this project is the PCB, so don't damage it. No removed parts will be reused for this project.

Remove: The ribbon cable and the LED board, SW1, SW2, JK1, SA1, T1, D1, D2, VR1, R1, R2, R3, R4, R5, R6, R8, R16, C4, C5, C6, C7, C8, C9, C10, C15, C20, IC1, IC5, X2. That is, everything from the large notch to the switch end of the PCB, plus IC1. There is no magic order to this. Just cut whatever you can get to most easily. Leave: IC2, IC3, IC4, X1, C1, C2, C3, C11, C12, the unmarked capacitor near R1, and the diode on the bottom of the PCB.



Next, remove some unneeded traces. Using a sharp knife, cut a section of copper from both sides of IC2 pin 17 on the bottom of the PCB: make a cut near the pin and another about 1/8 inch away. Work the knife under the copper and peel it up. Be careful to not let the knife slip and cut any of the other traces. Now, cut the trace at pin 16 and follow it to the pad near pin 14, cut it there, and peel it off. Repeat this with the trace at pin 9 on the top. Find 3 short traces, 2 narrow and one wider, between IC2 pins 1 and 28, going to solder pads. They come from pins 12, 9, and 10. Find the pads on the bottom and peel off the traces there. Also remove the wide trace running between them on the bottom of the PCB, running nearly the full width of the PCB, to make room to drill a hole later. File or sandpaper away any rough places in the newly bare areas, remove any solder bridges and generally clean up the PCB. Use paint thinner or alcohol to remove any flux. Lightly sand

the green paint on the top to let the glue adhere later. Just roughen it, don't sand through to the copper.

Modify the 9 pin RS232 connectors. Some connectors have pins that extend through the plastic base to be soldered into a PCB. Clip the pin ends that would normally extend through a PCB close to the plastic body of the connectors. Place the connectors on a flat surface and make sure the leads don't reach that surface. Mix some epoxy and spread it over the cutoff pins and the plastic body where it is recessed slightly. Lay it, epoxy side-up, on a plastic wrapped covered flat surface to set. Clamp or weight it down enough to keep it in place. Alternatively, use instant glue to hold a thin (.020) strip of plastic to the bottom of the connector body. Trim and drill the plastic strip to match the connector base after the glue sets. Some connectors don't have the pins run through the base. They are unsupported at the PCB end. Just bend the pins out straight and wire to them. You may clip them shorter to save space. The purpose of all this is to keep the pins from contacting any traces on the PCB and shorting out the signals, or letting the magic smoke out of an IC, while allowing easy connections to the pins.

Now modify the PCB. Since it isn't wired for +/- 12V, you must add 2 wires for that. -12V is easy, since pin 1 is on top. +12V is more difficult; pin 2 is on the bottom. A hole allows a wire to pass from the bottom to the top. Holes will be needed to mount the new RS232 connector: When you hold the PCB up to a bright light, you can see through where there is no copper on either side. Locate a clear area near pin 1, mark it and carefully drill a hole large enough to clear the insulation on a 30 gauge wire. Don't drill through any traces! An alternate method is to use a thin saw (like a coping saw) or a file to make a notch near pin 1 at the base of the large slot. Cut just deep enough to clear

a 30 gauge wire, and avoid the copper ground planes. Later, you can glue the wire in place. Place the PCB in the case bottom. Mount the 9 pin connectors on the cables. Lay the connectors in place on the end of the PCB to check the fit. You probably will need to widen the opening in the case slightly.

Use the nail clippers, diagonal cutters, and file for this. Save the latching tabs. If you are building only one port now, leave room for a second connector, or even mount two now to make adding a second port easier. The connectors can extend from the case, but keep the mounting holes far enough on the PCB for strength. Remember those connectors must support the cables. When the fit is acceptable, mark the PCB and drill clearance holes for your mounting screws. Oversized holes in the PCB let you move the connectors slightly, lining them up for a neater appearance. Peel away any traces the screws may touch, excepting ground. Separate the cables and connectors. Use a larger drill bit to deburr all holes, so you won't nick the insulation when you run wires through them. Be careful not to remove too much material. Mount the connector retainers (the "long nuts") then mount the connectors on the PCB. Trim the case halves to fit around them, bottom first, then the top. Finish the case now and take time to make it look good - this is the part everyone else will see. Maybe you can get help from a plastic model builder you know.

Mark the 1488 and 1489 ICs on the underside with their part number and pin 1 location. You can't read the tops after they're glued down! Use epoxy or instant glue to mount them with pins up between IC1/IC2 and the 9 pin connectors. Put them in line, pin 1 of each toward the 9 pin connector, with the 1489 nearest the ACIA. This keeps all the parts for one port in a row so they're easier to wire. Leave a little

room between rows (ports) for placing wires - 1/4 inch is enough. There will be lots of space for expansion in case you want to add something else to this pack.

You can now drill another hole near the end of IC2 to run wires through, you can just run wires around the edge, or you can connect to IC2's pins on top of the PCB.

If you have a favorite method wiring point to point, by all means use it. Leave about 1/4 inch of slack in each wire so you can move it out of the way for the later wires. Where I can, I wrap the wires onto the pins for a better connectic Strip about 1/4 inch and wrap as on wire-wrap pin. Strip both ends before connecting the first end. These pins are meant to be soldered, not wrapped, so be gentle. To connect to pins on IC2 and IC4, remove most of the solder, strip 1/8 inch of insulation, bend a hook in the bare end, squeeze it onto the pin, and resolder. Don't leave any bare wire to cause shorts. Wire to the 9 pin connectors the same way. Stagger the level of the wires between pins for more room. Wire to the inner/lower pins first so you can reach between the others to solder them without burning the wires on the outer/upper pins.

Run the new wires. Try to use unique colors for the connections between different parts. A widely used standard is to use red for +5V and black for ground. You may use only one color for all connections; electrons don't care, but that will make troubleshooting more difficult.

I don't recommend using a Y-cable. If you must, modify the following +/-12V and +5V wiring to use a separate power source, either a voltage converter, a wall plug, or a power supply. You can install a polarized plug and socket with at least 4 pins for +12, -12, +5, and ground. Connect your ground lead to any ground point, and use the following to wire the other voltages to the ICs. Use the pinout

Dynamic Systems Presents

Speedisk

Version 1.0

Single user version only \$99.95

Speedisk is a complete disk optimizer/defragmentator for OS-9/68000

- Runs under ALL versions of OS-9/68000, including ver 3.0
- Requires no modifications to system software (RBF)
- Understands all aspects of the file system including variable sector sizes, any cluster size, and linked files or directories
- Full screen display to monitor progress
- Intelligent file/directory placement to minimize the search time and reduce future fragmentation problems
- Over 3 years in development

All orders add \$5.00 for shipping. Checks or money orders only please. Call for details.

Dirt Cheap Computer Stuff Co.
 "Cheap but not trash"
 1368 Old Hwy 50 East
 Union, Missouri 63084
 314-583-1168

Tales from the Crypt

By Bob van der Roel

I've got a teenage daughter who is now using my computer...however, she is a bit hesitant about having all her "private stuff" on my hard drive. And she already knows that the password protected login system might keep her brother from reading her files...but not me. So I explained how one can encrypt a file so that no one can read it. She figured that was a great idea, and I went looking for the program. Of course, I couldn't find it, (well, I did find a couple, but the were inadequate for various reasons). After writing this, I figured that with communications being the theme of this issue, what better topic than the prevention of certain communications, (especially where teenage children are concerned).

There are many different methods of encrypting a file. As a matter of fact, entire books have been written on the subject. The code presented here can probably be broken by the CIA, but I don't think they care too much about my daughter's "private stuff".

Crypt uses a simple method to make the original message unreadable. It takes each character in the original message and EORs it with a character in a user supplied "key". To return the message to its original state, the characters are EORed a second time with the same key.

To make things a bit more secure, my version of the program does the EOR



twice for each character. This prevents a partial key from producing a partial decoding. The following program will work on OS9/68000 and OS9/6809 machines...and the files can be encoded on one system and decoded on the other. You can even use crypt to encode binary programs if you want (for example, those x-rated gifs you might have).

But a word of caution: The degree of security you achieve is directly proportional to the length of the key—the longer the key, the more difficult it is to decode the message; however, long keys may be harder to remember!

And if you forget the key you will not be able to decode the message. Also, the result of the encryption will not be transmittable as text on a BBS or other electronic service—you will have to transmit the file as "binary".

To use crypt, just pass the name of the file followed by the key as parameters. For example:

```
crypt diary secret
```

will encrypt the file "diary" using the key "secret". To return the file to its original format, just type the same command.

By using large block reads and over writing the existing file in place, cry works quite fast. I timed it on a 230K file on my MM/1 and it took 4 seconds to process the file.

If you'd like to comment on this program, or any other OS9 topics, drop me a note. I can be reached at:
PO Box 355, Porthill, ID 83853 or
PO Box 57, Wynndel, BC, Canada V0B 2N0 or
Compuserve 76510,2203.



of your power source to wire to it. If you provide a separate 5V source, cut the trace between C1 and the edge connector pin 9. Test your power supply before you connect it to your DCM Pak.

Wire +/-12 volts. The edge connector pin 1 is -12V, pin 2 is +12V. Pin 1 is marked on top, pin 2 is at the same end on the bottom. Cut wires long enough to reach from those pins along the side past X1 to the 1488 ICs, plus some extra. I use a felt tip marker to put stripes on these wires to help identify them later. Strip 1/16 inch, hold the bare end across the PCB end of the pin and solder. Use very little solder so it won't run up the pin. Don't touch the wire insulation until it cools or you will crush it and expose bare wire. Run the wire from pin 2 through the hole or slot you made. Connect the other ends, +12V to pin 14 of both 1488s, -12V to pin 1 of both 1488s. Leave plenty of slack so you can route these wires well away from the rest of the circuitry. Glue them to the PCB so they can't slip under a mounting screw.

Wire +5 volts. Use your ohmmeter or continuity tester to find the +5V pads; they are all over the board. Look for continuity to the side of C1 that connects to the edge connector pin 9. You will need to temporarily remove the aluminum foil from the edge connector - it connects all the pins together. Use pads that are close to the 1489s pins 14. The 1488s don't use +5V. Mark the pads to use, replace the aluminum foil, then wire.

Wire the ground connections. There are even more potential ground connection points than there were for +5V. Pick pads that are in narrow copper areas; they are much easier to solder. Connect a ground to the 9 pin connectors, the 1488s pins 7 and the 1489s pins 7. You may need a soldering iron with higher wattage to make good connections to ground.

Wire IC2, one 1488, one 1489,

and 9 pin connector (see tables). It is good idea to test this port now, while the ACIA is easy to replace. You will need a T3 descriptor - see below.

Bend up pins 2, 5 through 12, 16, and 17 of the new ACIA. Make them stick straight out from the case. Place the new ACIA on the old one, with both pins touching. Use 1 or 2 strips of thin cardboard between them as a spacer to allow air circulation. Bend the pins of the new ACIA as necessary until it sits on the old ACIA with all the pins touching. Recheck the orientation, then solder 2 corner pins (1 and 15). Check again, adjust if needed, then solder all touching pins together. Use a minimum of heat and solder, and take some time between pins to let things cool. Slip the cardboard strips out. Now wire the straightened pins of the new ACIA, the last 1488, 1489 and 9 pin connector (see below).



Wire the new crystal to the new ACIA. Trim the crystal leads short, wrap a 1 inch length of stripped 30 gauge wire on each lead and solder. Don't overheat. Insulate both leads. Wrap the other end of the wires to pins 6 and 7 of the new ACIA. Position the crystal on and fasten it to X1. Double sided tape or glue will keep it in place.

On the bottom of the PCB connect a wire from the old IC1 hole 8 to IC4 pin 1 and connect IC4 pin 2 to your new ACIA pin 2. If you are building only one port, you can change the address from \$FF6C to \$FF68, the same as the Deluxe RS232 Pak. Most if not all DECB program expect the ACIA at \$FF68, and you can use the stock T2 under OS-9. Cut a 1/16 inch section from the trace going to IC2 pin 2, wire IC2 pin 2 to IC4 pin 2, and wire

JWT Enterprises

Optimize Utility Set 1:

→ Optimize your floppies and hard drives quickly and easily! → Includes utility to check file and directory fragmentation.
 → Works alone or with Burke & Burke *repack* utility. → One stop optimization for any level 2 OS-9 system.
\$29.95; Foreign Postage, add \$3.00

Optimize Utility Set 2:

→ Check and correct any disk's file and directory structures without any technical mumbo-jumbo.
 → Run periodically to maintain the integrity of your disks as well as the reliability of your data
 → Especially useful before optimizing your disks.
\$19.95; Foreign Postage, add \$3.00

Optimize Utility Set Pac:

→ Get both packages together and save!
\$39.95; Foreign Postage, add \$4.00

Nine-Times:

In each issue:

- The bi-monthly disk magazine for OS-9 Level 2
 - Helpful and useful programs
 - C and Basic09 programming examples.
 - Hints, Help Columns, and informative articles
 - All graphic/joystick interface
 - Can be used with a hard disk or ram disk
- One Year Subscription, \$34.95;**
Canadian Orders, add \$1.00; Foreign Orders, add \$8.00

Back-Issues:

From May 1989, write for back issue contents
\$7.00 each; Foreign Orders, add \$2.00 each

Magazine Source:

Full Basic09 code and documentation for the presentation shell used with Nine-Times.
\$25.95, Foreign Orders, add \$5.00

NINE TIMES

Technical Assistance & Inquiries
 (216) 758-7694

JWT Enterprises
 5755 Lockwood Blvd.
 Youngstown, OH 44512



Copyright © 1992

OS-9 is a trademark of Microvare Systems Corp. and Motorola, Inc.

Foreign postage excludes U.S. Territories and Canada. These products for OS 9 Level 2 on the CoCo 3. Sorry, no C.O.D.'s or credit cards; Foreign & Canadian orders, please use U.S. money orders. U.S. checks, allow 4 weeks for receipt of order. Ohio residents, please add 6% sales tax.

```
(* convert dentry into a name *)
name:=""
IF dentry(1)>0 THEN
  FOR i:=1 TO 29
    EXITIF dentry(i)>127 THEN
      REM last character in filename
      name:=name+CHR$(dentry(i)-128)
  ENDEXIT
  name:=name+CHR$(dentry(i))
NEXT i
ENDIF

(* now we add it to pathlist in reverse order *)
pathlist:="/" + name + pathlist
ENDLOOP

(* comes here when we got root dir.
* get device name
regs.a:=path
regs.b:=$0E
regs.x:=ADDR(dentry)

RUN syscall($8D,regs)
CLOSE #path

name:=""
FOR i:=1 TO 32
  j:=dentry(i)
  EXITIF j>127 THEN
    name:=name+CHR$(j-128)
  ENDEXIT
  name:=name+CHR$(j)
NEXT i

pathlist:="/" + name + pathlist

(* if subroutine must restore current
* working directory
CHD pathlist
PRINT pathlist
END
```



-Mike Guzzi

HANGUP

BY: WAYNE CAMPBELL

This month's theme is communications, so I thought I'd take the opportunity to discuss the problem of illegal logoffs when a Basic09 subroutine module is in use.

As those of you who run (or have run) the StG Login Package (and used Basic09 subroutines to handle various user tasks, such as the original NR (News Reader) know, if a user logs off illegally (and a Basic09 subroutine is the current process), the I-Code modules are left hanging in memory and must be manually unlinked to kill them. You also know that trapping the process abort error (228) does nothing.

Well, I am now going to explain why this happens. Some of you may already know this, but, for those of us who are less enlightened, the information presented here may be of some use.

When a process is aborted, it means that OS9 received a signal. With user-defined signals, the signal must be intercepted by the process to avoid abortion. However, when a user logs off illegally, the signal sent is the kill process signal (signal 0), which is non-interceptable (almost). The exception is, when a process is in system state, the kernel prevents the abortion from taking place until the process returns to user state.

Now, the reason that the I-Code subroutines are left hanging in memory is because the only module being unlinked in a process abort is the PRIMARY module (in this case, RunB). The subroutines just stopped execution, and weren't even aware that an error occurred. And, since at least one of the subroutines in use was linked to the process (and was NOT unlinked), the module(s) were left in memory, unable to be killed.

I think that there may be a way to 'fix' this problem, though someone with more knowledge of the kernel would know more. It would involve a patch to (or re-write of) the kernel that would cause the system to suspend the process abort, check to see if the process' primary module is RunB, and then proceed. If the process' primary module is RunB, then the kernel would pass the process abort error code (228) to the process, wait for the process to abort the subroutines, and then proceed with the process abort in the usual manner.

Since RunB leaves it to its subroutines to check for errors, the subroutines would have to trap the 228 error code, or the procedure would just error out (and possibly get hung in an error loop if there was error trapping that didn't account for the 228 error).

If someone reading this (who can answer the question of patching the kernel) wishes to respond to that question, you can write to me, care of this magazine.

-Wayne Campbell

BASIC Training

by Jim Vestal

This month I present a Basic09 program submitted by Mike Guzzi, called BPWD, (thanks Mike!). It will return the path name of the current working directory, from within a Basic09 program. He modeled his program after the OS-9 command PWD.

Interesting to note that Wayne Campbell also submitted a similar program called PD and wrote an article explaining the theory behind finding the current directories (both the current working and execution directories). See his article for all the technical details.

Both BPWD and PD use similar methods to accomplish the same thing.

I enjoyed comparing the individual programming styles of both Mike and Wayne.

I hope that you too will take some time and look over both programs then decide which one you will use within your own projects.

Till next time,

-Jim Vestal

```
PROCEDURE bpwd
(* pwd command in basic09 *)
(* taken from disassembly of pwd *)
(* by Mike Guzzi for use of the OS-9 Underground *)
TYPE registers=cc.a.b.dp.BYTE; x.y.u.INTEGER
DIM pathlist:STRING[128]
DIM name:STRING[32]
DIM dentry(32):BYTE
DIM path:BYTE
DIM i,j,k:INTEGER
DIM regs:registers
DIM p_lsn(3):BYTE
```

```
DIM c_lsn(3):BYTE
DIM t_lsn(3):BYTE
(* start of code *)
pathlist=""
(* open current dir, get current LSN and parent LSN *)
OPEN #path,"":READ+DIR
GET #path.dentry
p_lsn(1)=dentry(30)
p_lsn(2)=dentry(31)
p_lsn(3)=dentry(32)
GET #path.dentry
c_lsn(1)=dentry(30)
c_lsn(2)=dentry(31)
c_lsn(3)=dentry(32)
(* this is the main loop
* compare LSN's if the same get device
* name and exit *)
LOOP
EXITIF p_lsn(1)=c_lsn(1) AND
p_lsn(2)=c_lsn(2) AND
p_lsn(3)=c_lsn(3) THEN
(* found root dir
ENDEXIT
CLOSE #path
CHD ""
(* now we must get the old "." entry name open
* parent and get new LSN's then find the name
t_lsn=c_lsn
OPEN #path,"":READ+DIR
GET #path.dentry
p_lsn(1)=dentry(30)
p_lsn(2)=dentry(31)
p_lsn(3)=dentry(32)
GET #path.dentry
c_lsn(1)=dentry(30)
c_lsn(2)=dentry(31)
c_lsn(3)=dentry(32)
LOOP
EXITIF EOF(#path) THEN
REM no more entries
PRINT "Error - Can't find dir entry"
CLOSE #path
END
ENDEXIT
GET #path.dentry
EXITIF dentry(30)=t_lsn(1) AND
dentry(31)=t_lsn(2) AND dentry
(32)=t_lsn(3) THEN
REM found dir entry
ENDEXIT
ENDLOOP
```

IC4 pin 1 to IC1 hole 8. You can move the wire at IC2 pin 2 and restore the original connection later, if you add a second port, by replacing the cutout section with a short wire.

Use the following table to wire the ACIAs to the 1488s and 1489s. Pins 9, 10, and 12 of IC2 are available at the solder pads between pins 1 and 28 where you removed traces on the bottom. Run the wires from IC2's other pins through the hole you drilled, or around the edge. Both ends of all the wires from the new ACIA will be on top of the PCB.

ACIA pin	1488 pin	1489 pin	Name
8	4+5	-	RTS
9	-	3	CTS
10	2	-	TXD
11	9+10	-	DTR
12	-	6	RXD
16	-	8	DCD
17	-	11	DSR
5	n/c (no connection)		

Above, 4+5 and 9+10 means connect those pins together and to the ACIA. Each ACIA connects to only one 1488 and one 1489. Go through the table for each ACIA.

Use this table to connect the 9 pin connectors to the 1488s and 1489s, and to check your cable wiring.

25 pin	9 pin	1488 pin	1489 pin	Name
22	9	n/c (no connection)		
5	8	-	1	CTS
4	7	6	-	RTS
6	6	-	13	DSR
7	5	Wire to any ground		GND
20	4	8	-	DTR
2	3	3	-	TXD
3	2	-	4	RXD
8	1	-	10	DCD

The table has the 9 pin connector pin numbers in reverse order to remind me to remind you to wire the lower row first.

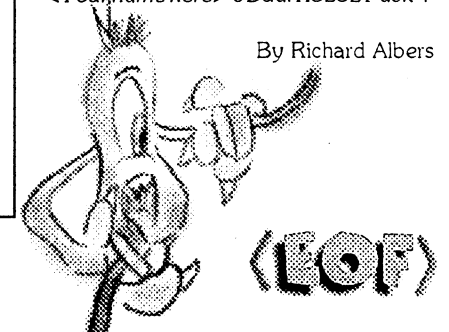
Again, each 9 pin connector is wired to only one 1488 and one 1489. Go through the table for each connector.

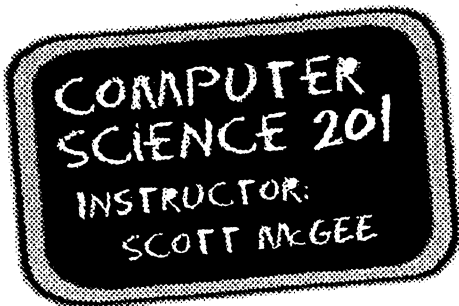
Check your work. Look for wir clippings, solder splashes, damaged insulation, shorts, poor connections, etc. Fix it now. You can hold the wires to the PCB with silicone glue/caulk. Solder all connections to make them permanent. Attach the shield with the 4 pins. You may need to trim it to clear the 9 pin connector mounting screws. Mount the PCB in the case with the four screws, remove the foil, and clean the 40 pin edge connector. Mark the 9 pin connectors as T2 and T3; T2 is connected to the new ACIA. Snap the case top on.

The T2 descriptor can be used as is, and a copy can be patched to provide a T3 descriptor. Since both ports will be in the same Multi-pak slot, you can't use the stock T3: it's for slot 2. Dump your T module. Look for two bytes with "FF68" and two bytes with "54B2". In my stock T2 they are at offset 000F+0010 and 002E+002F respectively. Patch them to be "FF6C" and "54B3", update the CRC with your preferred utility, and save as T3. Build a new OS9boot including T2 and T3.

Plug your new 2-port RS232 pack into slot 1 of your Multi-Pak. Without the aluminum foil! Now show your pride in your new creation: Cover that "DCM label with a new one proclaiming: "<Your name here>'s Dual RS232 Pack".

By Richard Albers





Lesson 3: Doubly Linked Lists

Instructor's Note:

Before I start today's lesson, let me say a word about homework. I realize that there are a wide variety of readers who will read this column. Each article contains a homework assignment. *I do not expect everyone to actually code the assignment.* That would be kind of silly if the reader was a long time programmer, familiar with the material, and just reading the article to see if I have something new or interesting to say.

On the other hand, if you are new to these concepts, I highly recommend that you *do* try to code the homework assignments. I also expect that some people may have problems with some of them. If you happen to fall into this category, please feel free to contact me by any of the means presented in each article, (email, US Mail), for help with your code.

I would love to have someone who is struggling with a new concept learn it because I was able to help. I won't be issuing any grades but if you are learning new concepts here, please do the homework. And please, ask me if you have any trouble with it.

Errata:

Last time, I made a typographical error in my discussion. At one point, I said: "if(list = NULL)"

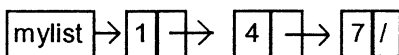
Unfortunately, this will ASSIGN a NULL to list!

It should have read:

"if(list == NULL)"

I hope this didn't inconvenience anyone. (oh, and an extra 10 points if you caught it!)

Previously, we have discussed Linked Lists. Today I wish to discuss a common variation of the linked list called the Doubly Linked List. A standard linked list looks like it following:



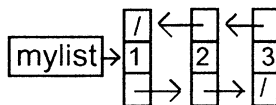
With a doubly linked list, in addition to the pointer to the next node, we have a pointer to the previous node. Thus, the declaration for it may look like:

```
typedef struct DLLIST *dllist; struct
DLLIST{ int data; dllist next; dllist
previous; }
```

Graphically, each node would look like this:



and a list would look like:



Here, "mylist" points to the first node, which contains one, (1). The "previous" pointer in that node is NULL, (indicated by the '/'), to show that it is the head of the list and that there is no previous node. In the last node, the "next" pointer is NULL, indicating the end of the list, while the "previous" points back up the list. You will recall the parent pointers we had to use with a standard

```
(*
(* If not the root dir
IF Parent_Sector<>Child_Sector THEN
REPEAT
(*
(* Open next dir
IF Data_Dir THEN
CHD ".\ "
OPEN #Dir_Path, ".":READ+DIR
ELSE
CHX ".\ "
OPEN #Dir_Path, ".":READ+EXEC+DIR
ENDIF
(*
(* Get '.' and '.' from current dir
GET #Dir_Path,File_Entry
(*
(* Determine sector numbers
Byte_Sector:=File_Entry(1).Sector_Byte_ID
Int_Sector:=File_Entry(1).Sector_Int_ID
Parent_Sector:=Byte_Sector*65536.+Int_Sector
(*
Byte_Sector:=File_Entry(2).Sector_Byte_ID
Int_Sector:=File_Entry(2).Sector_Int_ID
Child_Sector:=Byte_Sector*65536.+Int_Sector
(*
(* If '.' and '.' are the same then current
dir is root dir
IF Parent_Sector=Child_Sector THEN
Is_Device:=TRUE
ENDIF
(*
(* Get the entry for the previous dir
Child_Sector:=Temp_Sector
REPEAT
GET #Dir_Path,Check_Entry
(*
(* Determine sector number
Byte_Sector:=Check_Entry.Sector_Byte_ID
Int_Sector:=Check_Entry.Sector_Int_ID
Check_Sector:=Byte_Sector*65536.+Int_Sector
(*
UNTIL Check_Sector=Child_Sector
CLOSE #Dir_Path
(*
(* Set '.'
Last_Entry:=File_Entry(2)
(*
(* Determine sector number
Byte_Sector:=Last_Entry.Sector_Byte_ID
Int_Sector:=Last_Entry.Sector_Int_ID
Temp_Sector:=Byte_Sector*65536.+Int_Sector
(*
(* Get the name of the previous dir
Pointer:=0
Current_Name:=""
REPEAT
Pointer:=Pointer+1
Current_Name=Current_Name+CHR(AND(Check_Entry.Name
(Pointer),127))
UNTIL Check_Entry.Name(Pointer)>127
(*
(* Add name to pathlist
Path_List:="/" +Current_Name+Path_List
(*
(* If the root dir then add the device name
IF Is_Device THEN
Path_List:=Device_Name+Path_List
ENDIF
UNTIL Is_Device
(*
(* Now chx or chd back to original dir
IF Data_Dir THEN
CHD Path_List
ELSE
CHX Path_List
ENDIF
ELSE
(*
(* If the root dir then set pathlist to device name
Path_List:=Device_Name
ENDIF
(*
(* Print pathlist; To return Path_List to a calling
procedure
(* without printing, delete or REMark out this line
PRINT Path_List
(*
(* End of program
END
END
```

-Wayne Campbell

Email: "zog!wayne@abode.ttank.com"
or "os9under@AOL.com"



Know someone who's
not reading the
OS9 Underground?
Sign up a Friend and you
can benefit too!
Check your mail for Details

Tandy's Little Wonder

the most complete reference ever written for the Color Computer!

This 140 page softbound book contains:

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

ONLY \$25 (+ \$2.50 S&H)

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

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

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

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

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

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

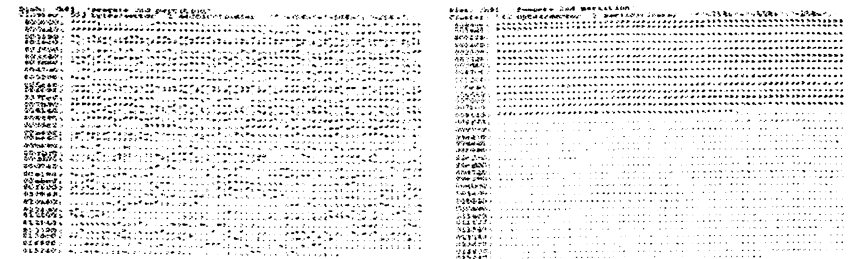
FARNA Systems PB

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

Defragment your OS-9 disk with Disk Squeezer.™

Frequent writings of small records of different files result in file contents scattered on the disk; this is called "Disk Fragmentation." A fragmented disk usually takes longer access times because the disk heads have to travel back and forth on the disk surface to read and write logically contiguous but physically fragmented file contents. Even worst, if a file is fragmented into too many pieces (4: for 256 byte sectors), you may not be able to extend the file size by even a single byte even though the disk has plenty of free space. This is called "File Fragmentation" and is fundamental problem with the RBF file manager.

Disk Squeezer automatically detects fragmented files and makes them more contiguous. Disk Squeezer also recognizes disk contents so that free sectors will be concatenated contiguously. This prevents future file fragmentation as well as disk access performance degradation. See the "before" and "after" graphical sector usage analysis of a real hard disk below.



Before Squeezing



After Squeezing
Disk Squeezer: \$295.00

Other OS-9/68K programs available from ARK Systems USA:

- UD-Cache II - Light Speed Disk Cache** \$149.00
- L\$rcDbg - Split Displays for Debugger and Application** \$50.00
- XSCF - Enhanced Line-Editing and Line-Recalling File Manager** \$60.00
- DDF - Idev Device File Manager** Coming soon
- PTF - Pseudo Terminal File Manager** Coming soon

*All programs work on any OS-9/680x0 system (V2.2-2.4). Fragmentation improvement factor may vary.

S&H: US (48 states) orders add \$4.00 for ground service or \$11.50 for FedEx 2nd day air; AK, HI and outside US ask for quotation. CA residents add 8.25%. Send your check or money order (no charge cards or CODs) with preferred disk format (important). 10% discounts for OS-9 User's Group members; send copy of your membership card.

ARK SYSTEMS ARK Systems USA
 P.O. Box 23
 Santa Clara, CA 95052
 Phone/Fax (408) 244-5358

linked list? Well, with a Doubly Linked List, this is no longer needed.

To do an insert into a doubly linked list, we do the following:

(Please note that I will now begin using pseudo code, rather than 'C' code)

```

if node = NULL list = new else
if node->data > target new->next = list
list->previous = new list = new else
node = list while node->data < target AND
node->next != NULL
node = node->next new->previous = node
new->next = node->next node->next = new
new->next->previous = new
    
```

Notice that this method handles an empty list by simply assigning the list to point at the new node. It handles insertion at the beginning of the list in a similar manner, but sets "new->next and list->previous" first. If the new node belongs somewhere after the first node, we enter the third section of the "if" clause. Here, we start by initializing node with list. After the while loop, node points at the last list item who's data is less than target. The next line lets new's previous pointer point back at this node. The following line sets new's next pointer to point to the next node in the list. At that point, "new" thinks it is in the list, but none of the list nodes know about "new". To fix this, we have to set the next pointer on node and the previous pointer on the next node. One thing to remember here is that, in the next to last line, we have just redirected node's next pointer to point to new. If we then set node->next->previous, we would be changing new's previous pointer. Since new->next now points to the next node, we need to use it instead.

Ok, now lets take a crack at deletion.

Again, I will do it in pseudo code. If you are new to pseudo code, remember, it is just a way to explain the process

without getting caught up in the syntax. Mypseudo code tends to look a lot like 'C' because I normally program in 'C'. I have seen other pseudo code that was straight English text with no symbols. The idea is to convey the meaning, not to be able to compile it!

Lets look at the following:

```

if list = NULL
quit
if list->data = target list = list->next
free list->previous
if list != NULL list->previous = NULL
else node = list
while node->data < target AND node->next != NULL
node = node->next
if node->data != target
node = NULL
if node != NULL
node->previous->next = node->next
if node->next != NULL
node->next->previous = node->previous
free node
    
```

This code looks more complicated than it is. Lets take it a bit at a time. First, we check for an empty list. We'll just bail out if we have one.

```
if list = NULL, quit
```

Next we have to handle the case of the node to delete being the first node. This is a special case since we have to modify the pointer to the list. The first thing we do is to step the list pointer past the node. We can do this without having a pointer to the node because we have the previous pointer to use. Using that, we then free the node.

```
if list->data = target list = list->next free list->previous
```

Now, we must update the previous pointer to show that the node is now the head of the list by setting it to NULL.

```
if list != NULL list->previous = NULL else node = list
```

Finally, if the node is not at the head

```

IF Exec_Dir<>"-X" AND Exec_Dir<>"-x" THEN
  ErrorMessage:= "pd: unknown option"
  END ErrorMessage
ENDIF
(* If parameter then check execution dir
Data_Dir:=FALSE
(* Go here if no parameter
1 er:=ERR
ON ERROR
(* Assume not root dir
Is_Device:=FALSE
Path_List:= ""
(* Open initial dir
IF Data_Dir THEN
  OPEN #Dir_Path,"":READ+DIR
ELSE
  OPEN #Dir_Path,"":READ+EXEC+DIR
ENDIF
(* Get device name for current drive
Call_Code:=$8D
Regs.A:=Dir_Path
Regs.B:=$0E
Regs.X:=ADDR(Device_Buffer)
RUN SysCall(Call_Code,Regs)
(*
(* Set device name
Device_Name:= ""
Pointer:=0
REPEAT
  Pointer:=Pointer+1
  Device_Name:=Device_Name+CHR$(AND(Device_Buffer(Pointer),127))
UNTIL Device_Buffer(Pointer)>127
(* Get '.' and '..' from current dir
GET #Dir_Path,File_Entry
CLOSE #Dir_Path
(* Set '..'
Check_Entry:=File_Entry(1)
(* Set '.'
Last_Entry:=File_Entry(2)
(* Determine sector numbers
Byte_Sector:=Check_Entry.Sector_Byte_ID
Int_Sector:=Check_Entry.Sector_Int_ID
Parent_Sector:=Byte_Sector*65536.+Int_Sector
(*
Byte_Sector:=Last_Entry.Sector_Byte_ID
Int_Sector:=Last_Entry.Sector_Int_ID
Child_Sector:=Byte_Sector*65536.+Int_Sector
Temp_Sector:=Child_Sector
    
```

MultiBoot by Terry Todd & Allen Huffman

Now have up to SIXTEEN bootfiles on your startup disk!

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

OS-9 Req: CoCo 3, OS-9 Level 2.....\$19.95

Towel by Allen C. Huffman

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

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

OS-9 Req: CoCo 3, OS-9 Level 2.....\$19.95

OS/K Req: MM/1 or K-Windows Compatible.....\$24.95

992 CoCoFest SIMULATOR by Allen C. Huffman

Graphics "adventure" based on the 1992 Atlanta CoCoFest

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

OS-9 Req: 512K CoCo 3, OS-9 Level 2, 490K Disk Space.....\$ 9.95

OS/K Req: MM/1 or 100% K-Windows Compatible.....\$14.95

Send US funds plus \$2.50 shipping to:

Sub-Etha Software

P.O. Box 152442
Lufkin, TX 75915
(815) 748-6638

More items available! Contact us for a complete product listing!

the same. PD assigns the Device Name to the Path_List String, prints the string, and exits.

If it is *not*, PD begins a REPEAT loop that CHDs (or CHXs) back one directory at a time until it reaches the root directory. With each pass, PD keeps track of the previous directory so that it can find the filename for that directory when it retrieves the parent and child entries of the current directory. It then adds that filename to the Path_List string.

Once it reaches the root directory, it assigns the Device Name to the Path_List string, prints the string, CHDs (or CHXs) back to the starting directory, and exits.

If you were just going to use it as a means of remembering which data or execution directory you are in from the shell, it is useless. As soon as the process ends, the paths established in the shell's process are reinstated.

The routine for CHDing or CHXing back to the starting directory *does* serve a useful purpose.

But, what if you wanted to use PD in a program by calling it? If that little routine wasn't there, you would find yourself in the root directory! The reason is, Basic09 (or RunB) is the process, *not* your program. So, Basic09 (or RunB) is going to assume that the last CHD (or CHX) done is the directory you want to be in. PD would return a string stating that your current data path, (for example), is /D1/TEST/WIDGIT, if your program was to do a 'SHELL "dir"' at that point, you would find yourself in the root directory of /D1. In order to avoid this, PD *must* CHD (or CHX) back to its starting directory.

The above reference to calling PD from another procedure requires a minor change to the code:

Change the statement:

```
DIM Path_List:STRING[255] To:
PARAM Path_List:STRING[255]
```

If you don't want PD to print the

result, delete the statement: PRINT Path_List

Use the following syntax for calling PD from within your programs:

```
For Current Data Directory:
    RUN PD(<PathList>)
For Current Execution Directory:
    RUN PD(<PathList>,"-x") or
    RUN PD(<PathList>,"-X")
```

<PathList> is a string variable of up to 255 characters.

Finally:

If anyone can figure out the exact system call required (and the exact code to implement it) for OSK, send it to me, care of this magazine, or send me e-mail with all of the pertinent information.

Enjoy!

```
PROCEDURE pd
(* pd - print current data/execution directory path
* . Copyright (c) 1993 Wayne Campbell
* usage:
* pd - print current data directory path
* pd ("-x [or -X]") or
* pd -x [or -X] - print current execution directory path
(* Increase the length of Path_List if you run into problems *)

TYPE REG=CC,A,B,DP:BYTE; X,Y,U:INTEGER
DIM Regs:REG
TYPE ENT=Name(29),Sector_Byte_ID:BYTE;
Sector_Int_ID:INTEGER
DIM File_Entry(2),Check_Entry,Last_Entry:ENT
DIM Device_Buffer(32),Dir_Path,Call_Code,
Byte_Sector:BYTE
DIM Pointer,er,Int_Sector:INTEGER
DIM Parent_Sector,Child_Sector,Check_Sector,
Temp_Sector:REAL
DIM Is_Device,Data_Dir:BOOLEAN
DIM ErrorMessage:STRING
DIM Device_Name:STRING[33]
DIM Current_Name:STRING[29]

(* To return Path_List to a calling procedure, change DIM to PARAM
DIM Path_List:STRING[255]
PARAM Exec_Dir:STRING[2]

(* Assume check current data dir
Data_Dir:=TRUE

(* Check for parameter
ON ERROR GOTO 1
```

of the list, we point a temporary pointer at the head, and then use the while loop to step down the list until we either find the location of the node to delete, or reach the end of the list. The while loop will end when the we reach the end of the list because node->next is NULL, or when node->data is no longer less than target.

```
while node->data < target AND node->next != NULL
    node = node->next
    if node->data != target
        node = NULL
    if node != NULL
        node->previous->next = node->next
```

Since node->data may or may not be equal to target, we use an additional test to set node to NULL if node->data is greater than target because we only got here if the last node was less than target, so if this one is greater, we know that target is NOT in the list.

At this point, if node is NULL, we know that target was NOT in the list. If

not, then node is indeed the one we want. First thing we do from here is to set the next pointer on node->previous to the value of node->next to point it around node. Then we have to see if there is a node = node->next. If so, we need to set it's previous pointer to point back at node->previous.

```
if node->next != NULL
    node->next->previous = node->previous
```

Finally, now that we have isolate node, we must free it.

```
free node
```

For your homework assignment, re-write the pseudo-code presented her in your favorite programming language and add the needed code to create and initialize new nodes.

Next time, we will use a very similar type definition for a very different kind of data structure... trees!

Class dismissed!



-Scott McGee

Moving?

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

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

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

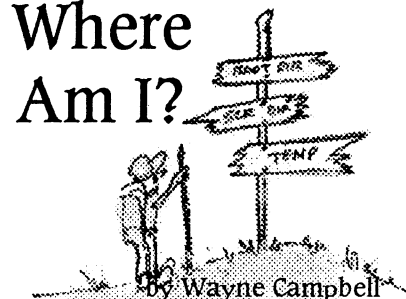
68XXX COMPUTER PRODUCTS from **Peripheral Technology**

- a company with a reputation for quality!

PT68K4-16, 1MB \$299.00
PT68K2-10, 1MB \$199.00
ALT86 for PC
Compatibility \$199.00
Profess. OS9 \$299.00

1480 Terrell Mill Rd. #870
Marietta, GA 30067
404/973-2156

Where Am I?



By Wayne Campbell

After I became familiar with the directory structure of OS-9, I found myself wanting to know how the commands PWD and PXD worked. Only now, after 4 years and upgrading to a CoCo3 and Level 2, have I learned how they work. This is especially important to me as a programmer, because there are times I would like to display the current data, (and/or execution), directory path in my Basic09 programs, but previously had to use, (in my opinion), inefficient code to do it.

Note to OSK users:

PWD and PXD use a system call to retrieve the current device name. OSK users will have to figure out how to do the appropriate system call to get the device name.

PD:

This procedure is a Basic09 PWD/PXD utility all in one program. To use it:

PD with no options returns the Current Data Directory Path. PD with the option "-x" (or "-X") returns the Current Execution Directory Path. PD with any other option returns the error message: "pd: unknown option"

PD was developed on a Tandy Color Computer 3 under OS-9 Level 2. It *SHOULD* run on any OS9 Level 1 or Level 2 platform.

How it works:

First, the Specified, (Current Data or Execution), Directory is opened for read,

and the parent, (..), and child, (.), entries are read. The system call to retrieve the Current Device Name is performed, and the current directory is closed.

The system call is a GetStat (\$8D) call: (Excerpt from the Level 2 manual)

SS.DevNm (Function Code \$0E)

Returns a Device Name

Entry Conditions:

A = Path Number (Path to currently opened file)

B = \$0EX = Address of 32-byte buffer for name

Exit Conditions:

X = Address of buffer, name moved to buffer

I suggest NOT using a 32-character string as the buffer (though it is leg: because Basic09 terminates any string using less than its maximum allocation with \$FF as the first character after the last character of the string. Example:

```
name$:="DD" (name$ is 32 characters in length)
```

```
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
```

```
-----
D D .(the rest is garbage) -----
4444FF
(the rest is garbage)
```

This is what name\$ would contain if used to receive the results of the call.

```
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
```

```
-----
44C4(the rest is garbage) -----
D D
(the rest is garbage)
```

As you can see, the Device Name would contain garbage, as there's no end-of-string marker. This is why PD uses a 32-byte array to get the name, and then build the string. Notice that SS.DevNm does NOT include a leading slash '/'. This must be added to the string.

Next, PD determines if the current directory is the root directory. In the root directory, '..' and '.', have the same LSN. If it IS, PD assigns the Device Name to the Path_List string, prints the string, and exits.

If the parent and child directories are

Northern Xposure

'Quality Products from North of the Border'

Smash! CoCo3 Level II Acade Game \$25.00

TheXder:OS9 \$29.95

(Send manual or rompak to prove ownership)

Matt Thompson's SCSI System \$25.00

Rusty Launch Basic Progs from OS9 \$20.00

OSTerm 68K v2.0 for the MM/1 \$50.00

Switch to OSTerm 68K from TasCOM \$30.00

(Send manual or disk to prove ownership)

OS-9 68K Software SC/ISpell/Rogue + \$5.00

Write for a free catalogue

● US Funds. Check or M.O. Prices include S&H ●

Alan DeKok

7 Greenboro Cres
Ottawa, ON
Canada K1T 1W6
(613)736-0329

Colin McKay