

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 offering 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 from 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
PAN UTILITIES

Distributor of Microware Systems Corporation Software

delmar co

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

The "International" OS9 Underground.

A Fat Cat® Publication

Magazine Dedicated to OS-9/OSK Users Everywhere!

The Future of the OS-9 Users Group

- Commentary on the User Group
- Finding malloc() Problems
- Mass Quantities - Disk Back ups

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
MMI1 Version \$39.95

WPShel

A Word Processing Oriented Point and Click Shell for all your word processing needs. Requires WindInt from your Multi-View 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 MMI1 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
MMI1 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

Announcing

4th Annual Atlanta CoCoFest

Holiday Inn, Northlake

October 2 & 3, 1993

Show Hours:

Sat Oct 2 10:00 AM - 5:00 PM
Sun Oct 3 10:00 AM - 3:00 PM

Vendor setup:

Fri Oct 1 6:00 PM - 11:00 PM
Sat Oct 2 8:30 AM - 9:45 AM

Admission : \$5.00 (Whole Show)

Reservations:

Holiday Inn, Northlake, GA
(800)-465-4329 or (404)-938-1026

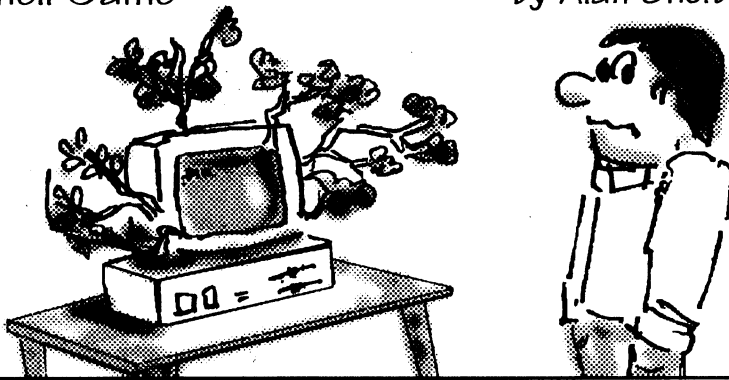
Sponsored by:

Atlanta Computer Society

PO Box 80694
Atlanta, GA 30366
BBS: (404)-636-2991

Shell Game

by Alan Sheltra



Smedley has definitely let his Directory Tree get way outta hand.

Advertisers Index

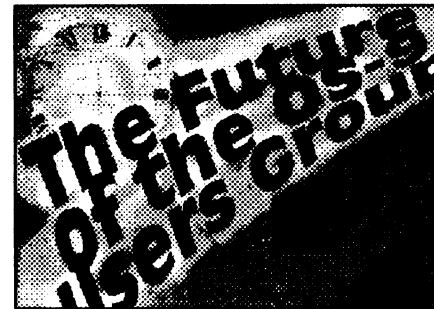
Vendor	Page
ColorSystems	IFC
CoNect	5
Computer Design Services	7
BlackHawk Enterprises	7
FAT CAT Publications Bookshelf	10
Bob van der Poel Software	10
JWT Enterprises	14
ARK Systems USA	17
OS9 Underground	20,37
Northern Xposure	21
Farna Systems	24
Sub-Etha Software	23
AniMajik Productions	28
Dirt Cheap Computer Stuff Co.	29
09-Online	33
Peripheral Technologies	36
Atlanta Computer Society	IBC
DELMAR Co.	BC

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

The "International" OS9 Underground® Magazine

Dedicated to OS-9/OSK Users Everywhere

Volume 1, Issue 10
C O N T E N T S



The Underground Staff

Editor/Publisher:
Alan Sheltra

Assistant Editors:
Jim Vestal
Steve Secord

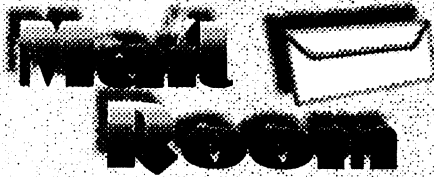
Technical Editor:
Leonard Cassady

Contributing Editors:
Wes Gale
Scott Griepentrog
Allen Huffman
Scott McGee
Eric Levinson
Bolsy Pitre
Bob van der Poel

Typesetting/Layout/Artwork:
AniMajik
Productions

Mall Room (Letters to the Editor)	4
Under It All (Editor's Column)	6
"The Future of the OS-9 Users Group" Commentary by David M. Graham	8
"Computer Science 201" "Lesson 2: Linked Lists, Insert/Delete" by Scott McGee	11
"Mass Quantities" "Making Back-up Disks" by Eric Levinson	15
"C Software Engineerin'g" "Standard C - Part II" by Leonard Cassady	18
"BASIC Training" "Tic Tac Toe" by Jim Vestal	25
"Finding malloc() Problems" by Bob van der Poel	30
"Writing a Device Driver - Part 3" by Bolsy G. Pitre	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.



Lookin' Good...

The magazine is really looking good...keep up the good work!

(Left on Fat Cat's answering machine!)

-Tony Podraza, Chicago, IL

Wants a New Standard

Enclosed you will find a photo-copy of a calendar that I thought you would like....



(Caption reads: "Wait a minute! The monster seems confused, disoriented. I think he's gonna passout! Get the nets ready!")

... This is a more specific question to all System IV owners. Under DataDex using VGA terminfo, the program shows codes like [7m[23m0H0H] at each line and before each letter that I input. What goes? This also occurs with VED under all except ABM85H.

Would that I could use only 1 term environment for all of programs.

Is there any chance of standardizing video out for the entire OSK world or some sort of library of sub-programs so that these different machines can run all the same programs without a major rewrite?

Thanks for the good & informative mag & keep up the good work!

-Tom Farrow - Denver, CO.

Just Zippin' Right Along

Just a fast note to you - You had a misprint on my Zip Code this issue - It WAS printed as 1934...

Just in case, I'll give you my full address here. And THANKS for a GOOD Mag that's getting Better.

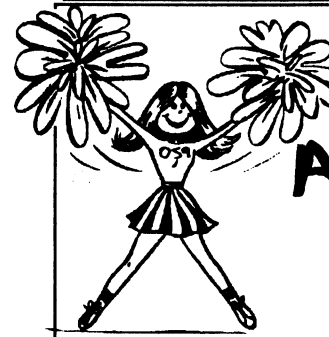
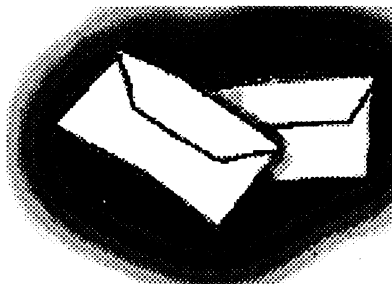
-John Baer - Philadelphia, PA.

Letters to the Editor can be mailed to:
The OS9 Underground Magazine
Letter to the Editor
4650 Cahuenga Blvd., Ste #7
Toluca Lake, CA 91602

Or Emailed to:

ZOGster@AOL.com (America Online)

ZOGster@delphi.com



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
- Fama 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... The Communications Issue

The Write routine is very much the same as the Read routine with the exception of the second assembly code line. One byte is moved from the address into d0. d1 is cleared and the routine returns to the file manager.

The TERM Routine

```
*****
* Term: Terminate A/D processing
*
* Passed: (a1) = device descriptor pointer
*         (a2) = static storage
*         (a4) = current process descriptor ptr
*         (a6) = system global data ptr
*
* Returns: none
*
* Error Return: (cc) = carry set
*              d1.w = error code
*
Term: move.l V_PORT(a2),a0 move the address into a0
moveq #$FF,(a0) write $FF into the port address
clr.w d1 return without error
rts
ends
```

The Term routine resets the A/D hardware by writing an \$FF to the port address. The ends directive marks the end bound of the psect, and we have an almost complete driver.

Next month we'll finish our coding by adding the GetStt and PutStt routines to implement sound sampling and playback. In the meantime, I would like to hear your comments, suggestions and questions regarding this series of articles. Feel free to contact me via U.S. Mail or electronic mail:

US Mail: PO Box 523, Waukee, IA 50325
 Internet: boisy@os9er.waukee.ia.us
 Delphi: BOISY

-Boisy G. Pitre



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

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 froms 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.

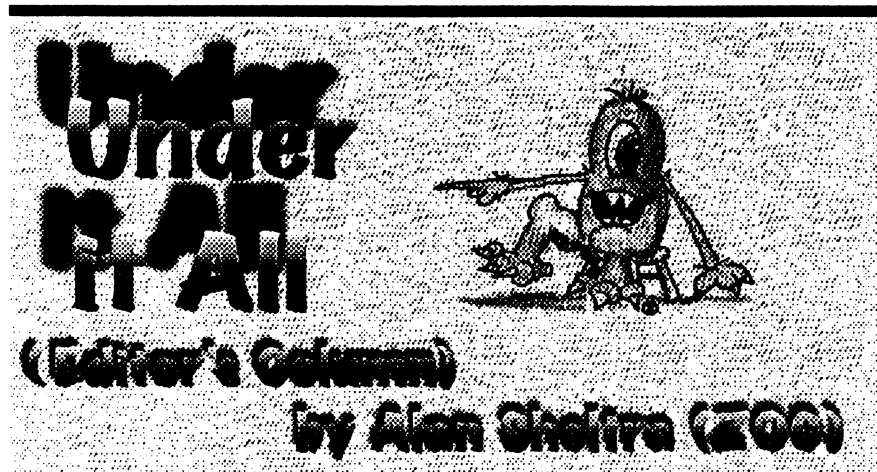
The no-slot RS232 port is a virtual clone of the mini-rs232 described above, and saves not only a slot but quite a bit of room in the finished package.

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



Needs Your Support

The OS-9 User Group will hold a conference at the Atlanta Fest (Oct 2 & 3) which will decide the by-laws and direction the new group will take. From much of the preliminary input I have seen regarding the new group's by-laws, this group should be able to avoid any potential problems like this last fiasco with Jim Destafeno.

Also, at the Atlanta Fest conference, a new User Group President and staff will be elected.

We need this group! Give them your support! The address to send your member fees (\$25.00) to are:

OS-9 Users Group, Inc.
6158 W. 63rd St. Suite 109
Chicago, IL 60638

Fat Cat Publications is making available, reprints of the MOTD (3 Issues from 1992 Boisy-era) available to the general public. These issues are 8 1/2" x 11" and come bound in plastic sheet protector. See the ad on page 10 for details. All profits from this re-print will go directly to the new User Group.

Speaking of publications, Fat Cat is planning it's first book due out for Christmas. Watch for announcements soon!

Atlanta Fest and Back Issues

By the time you read this the Fest is already in progress or just over with. I have been very busy getting back issues ready for the fest and making all the needed preparations. It's the reason we are a few days behind in shipping this month. Hope you have (or had) a chance to stop by the Underground Booth.

Back issues are now available from issue # 1 to the present issue at the cover price. Information will be mailed to all subscribers within a week to 10 days after receiving this issue as to pricing and availability.

The Underground and Fat Cat Publications hopes that the Atlanta Fest is a big success for vendors and promoters alike and great time for all who attend. We plan to have coverage of the fest in our next issue.

Call to SysOps...

Next month is out Communications issue and we are asking all sysop send your BBS info to us so we can include you in our free listing. Include the BBS name, SysOp's name, phone, hours, baud rate(s), parity and systems supported. If you are on a network, please include that as well.

•EOF•

The Q & A Column returns next month. We just plain ran outta space in this issue! See you next month.

-Alan Sheltra (ZOG)

file manager (SCF) does use a portion of the static storage for its own needs.

The branch table, which is referred to in the psect directive above, has seven entries. All but one are currently implemented in OS-9.

The INIT Routine

```
* Init
* Initialize A/D Port
*
* Passed: (a1) = device descriptor address
*         (a2) = static storage address
*         (a4) = process descriptor ptr
*         (a6) = system global data ptr
*
* Returns: nothing
*
* Error Return: (cc) = carry set
*              d1.w = error code
*
* Destroys: (may destroy d0-d7, a0-a5)
*
* This routine does the following:
*
* - clears port to eliminate line noise present
* when the computer is started.
```

```
Init move.IV_PORT(a2),a0 move the address into a0
clr.b (a0) clear the port
clr.w d1 return without error
rts
```

Remember from the article last month that the A/D port is initialized by writing byte \$0 into it. To do this, we need to load a register with the address of the device. The V_PORT constant evaluates to an offset within the static storage where the device address is stored. By moving this address into a0, we can use clr.b to write a \$0 there. We then clear register d1 (which clears the carry bit in the condition codes register) to signify that we've exited the routine without an error.

The READ Routine

```
* Read: Return one byte of input from the A/D port
*
* Passed: (a1) = Path Descriptor
*         (a2) = Static Storage address
*         (a4) = current process descriptor
*         (a6) = system global ptr
*
* Returns: (d0.b) = input char
*
* Error Return: (cc) = carry set
*              d1.w = error code
*
* Destroys: a0
*
Read: movea V_PORT(a2),a0 move address into a0
move.b (a0),d0 read byte into d0
clr.w d1 return without error
rts
```

In the Read routine, we move the port address into a0 and move whatever data is there into d0. d1 is cleared and we return to the file manager.

The WRITE Routine

```
* Write
* Output one character to the A/D port
*
* Passed: (a1) = Path Descriptor
*         (a2) = Static Storage address
*         (a4) = current process descriptor ptr
*         (a6) = system global data ptr
*         d0.b = char to write
*
* Returns: nothing
*
* Error Return: (cc) = carry set
*              d1.w = error code
*
Write:
movea V_PORT(a2),a0 move address into a0
move.b d0,(a0) write byte into port
clr.w d1 return without error
rts
```



by Boisy G. Pitre

In this installment we will begin writing the driver for the dual A/D convertor described last month. As I annotate the code, you should have the past two month's articles in front of you for reference, as well as a 68000 assembly language manual.

Driver Implementation

At the very least, we want our driver to be able to read from and write to the hardware one byte at a time. This fits nicely with the Read and Write routines of the OS-9 driver model. However, our A/D hardware contains valid data at any given time. This is not the case with serial and printer drivers which obtain valid data only when it is ready to be read.

Because data is always valid, our driver doesn't need to support functions like SS_Ready or SS_SSig. Likewise, programs that read from the A/D ports will never block. While this makes our coding much easier, certain elements of driver writing (such as the interrupt service routine and other status routines) will not be presented.

Now that we are aware of the semantics of this particular driver, let us examine the source code. Notice that for each routine, I've supplied the register calling and return conventions in comments.

The Psect Directive

```

Edition equ 01 current Edition number
Typ_Lang set (Drv<<8)+Objct
Attr_Rev set ((ReEnt+SupStat)<<8)+0
psect RMADDrv,Typ_Lang,Attr_Rev,Edition,0,ADEnt
    
```

The psect line supplies several necessary values which will be used in creating the final OS-9 object code upon linking. First, we give this section of code a name — RMADDrv. Second, we must define the type of OS-9 module we are creating. Since this is a driver, we combine the Drv and Object values here. Third, we use the ReEnt and SupStat values to indicate our driver is reentrant and must run in supervisor state. Next is the driver's edition number, stack size (which we set to 0) and finally, the entry point into the driver.

Static Storage and the Entry Table

```

* Static Storage (not used in this driver)
vsect
ds.b 1
endsect

pag
*****
* Branch Table
*
ADEnt dc.w Init
      dc.w Read
      dc.w Write
      dc.w GetStat
      dc.w PutStat
      dc.w Term
      dc.w 0 Exception handler entry (0=none)
    
```

Although 1 byte of static storage is reserved, our driver will not be using any for its own storage. Do keep in mind that the

NEW 68020 COMPUTER BOARD!!!

The CD68X20 *sizzles* at 25MHZ - processing the most complex calculations in a *flash!!!*

CD68X20-25, OK RAM	\$699.00
Professional OS9/020 V2.4	\$499.00
ULTRA C Compiler for OS9	\$299.00
SCULPTOR V1.14:6 for Business Software Development	\$ 79.00

Systems Available!

Computer Design Services
 2550 Sandy Plains Rd. Ste. 320-234
 Marietta, GA 30066
 404/973-2170

Paint version 1.0 IMS \$54.00 + 2.50 S/H

The commercial version of the Paint Program included with all MM/1 systems. Now includes file I/O for GIF and BRUN files which can be compiled to FLI animation with tools from the PIXutils disk. This introductory price won't last long!

Midi Paddle Boards. IMS \$75.00 + \$2.50 S/H

These high quality MIDI paddle boards are now in stock at our location, ready to ship immediately. Price includes cable and software for your MM/1 computer!

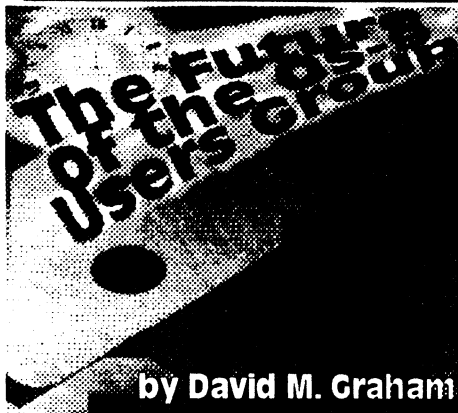
Serial Paddle Boards. IMS \$49.95 + \$2.50 S/H

Manufactured inhouse with boards supplied by IMS. These fine units come with cables and installation instructions, and are available for immediate shipment to interested MM/1 owners!

Call for availability of other products!

BlackHawk Enterprises P.O. Box 10552 Enid, OK 73706-0552

Call (405) 234-2347 from 9 am to 2 pm Central Time



by David M. Graham

It seems to be a common topic lately. The Future of the OS9 Users Group. Think about it. Of course most of us haven't, have we? In fact, of 245 ballots sent out by the UG officers earlier this year, only four were returned, four! Less than 3% of this group were committed enough to even vote, much less attempt to contribute in any meaningful way to a group that could, and should, act as a focus for the entire spectrum of the OS9 Community. Of course the census of members should tell you something too. A Users Group formed to represent all OS9 users in the United States, The ONLY Users Group to represent all OS9 users in the United States of America, and it has only 245 members?

Excuse me? OS9 is a prime mover in industry totaling a half BILLION dollars in sales each year. Count them dollars, \$500,000,000. Yes, indeed, that is 8 zeroes there, NINE digits. I suspect we all can agree now that the OS9 Users Group failed, even in its earlier incarnations, in obtaining serious penetration into the 'market' for OS9 users. And I suggest we must now consider why this is so.

User Group a Failure!

I submit to you - that the OS9 Users Group has largely failed to provide programs

that serve the needs of significant portions of the OS9 Community. that the OS9 Users Group has largely existed to support modern using users of OS9 machines, that major programs have simply been left to rot on the vine because the modem-based users making the decisions consciously chose to discriminate against those who have not got the money to use an on-line service, that an organization that was efficient and lasting was not to be found. Now, I believe that when Boisy Pitre sat down and thought about reforming the Users Group last year that he thought the same thing. I doubt that he would disagree with me one bit. I firmly believe that Boisy intended to set these problems straight! Then came Jim De Stefano. Why Jim?

Why Jim?

Well, the answer appears in my first paragraph. The answer is plain and simple. Jim De Stefano did NOT cause the current problem in the Users Group! "Wait!", you cry!, "What are you saying?. That low down skunk was personally responsible for this whole mess! He is the one that shut us down, redistributed the treasury, and did not even give us a chance to reorganize!" Well, could it be that we had a chance, could it be that we shut ourselves down, could it just be that the problem that allowed the Users Group to come to this low point is YOU and ME?

Well, I tell you neighbors, it most certainly IS! You, the Membership of the OS9 Users Group, did not care enough to nominate or vote for a candidate. You did not care enough to stand for election. You shunned your obligation to serve. Yes, SERVE, for if all are to be served, all must provide service! You've heard enough about you, you say? What did I contribute, you ask? How did you fail? Well, friends, much as it shames me to admit it, I never

```

memcheck_free(ptr,fn,ln)
void *ptr;
char *fn;
int ln;

{
if(!ptr) goto EXIT; /* ignore deallocation NULL ptrs */
if(!logfile) openlog();

fprintf(logfile,"free $%8x, %s() %d\n",ptr,fn,ln);
if(cnt!=-1)
{
int i;

for(i=0;i<cnt;i++) if(p_tab[i].ptr==ptr) break;

if(i>=cnt)
{
printf(logfile," Pointer not allocated in table\n");
}
else
{
p_tab[i].ptr=NULL;
amount-=p_tab[i].size;
}
while((cnt>0) && (p_tab[cnt-1].ptr==NULL)) cnt--;
}
free(ptr);
fflush(logfile);

EXIT;
}

openlog()
{
char buff[200];

sprintf(buff,"%s:%d",LOGFILE_NAME,getpid());
logfile=fopen(buff,"w");
if(!logfile) terminate("Can't create memory logfile");
}

#endif ifdef MEMCHECK
    
```

LISTING 2

```

/* memcheck.h */
#ifdef MEMCHECK
#define free(x) memcheck_free(x, FILE __LINE__)
#define malloc(x)
    
```

```

memcheck_malloc(x, FILE __LINE__)
#define calloc(x,n)
memcheck_calloc(x,n, FILE __LINE__)
#define strsave(x)
memcheck_strsave(x, FILE __LINE__)

void *memcheck_malloc(), *memcheck_calloc();
char *memcheck_strsave();

#endif
    
```

-Bob van der Poel



09-Online Systems

US Robotics Sportster 14,400 Modem External Version CCITT V.32 bis with V.42 bis throughput up to 57,600 bps.

Special sale price: \$250.00 (add \$5. for UPS shipping)

Send your paid order (check/M.O.), or inquiries to:

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.

NOTE: In openlog() terminate() is called if the file cannot be opened.

This is a standard routine I have in most of my programs. You might want to replace it with exit(). I use terminate() since it does some cleanup of the terminal, etc.

KNOWN BUGS: realloc() is not supported.

```

*/
#include <stdio.h>
#include <malloc.h>

#define PTAB_SIZE 1000 /* max number of entries */
#define LOGFILE_NAME "memorylog"

static struct
{
    int size; /* size of allocation */
    void *ptr; /* pointer to allocation */
} p_tab[PTAB_SIZE];

static int cnt=0; /* max entry number */
static int amount=0; /* number of bytes allocated */
static FILE *logfile=NULL;

char *
memcheck_strsave(data,fn,ln)
char *data;
char *fn;
int ln;
{
    register char *ptr;
    ptr=(char*)strsave(data);
    update_record(ptr,strlen(data),fn,ln,"strsave");
    return ptr;
}

void *
memcheck_calloc(numel,size,fn,ln)
int numel,size;
char *fn;
int ln;
{
    register void *ptr;

    ptr=calloc(numel,size);
    update_record(ptr,size*numel,fn,ln,"calloc");
    return ptr;
}

void *
memcheck_malloc(size,fn,ln)
int size;
char *fn;
int ln;
{
    register void *ptr;

    ptr=malloc(size);
    update_record(ptr,size,fn,ln,"malloc");

    return ptr;
}

static
update_record(ptr,size,fn,ln,func)
void * ptr;
int size;
char *fn;
int ln;
char *func;
{
    if(!logfile) openlog();
    fprintf(logfile,"%s %d bytes at %x, %s)\n",func,size,ptr,fn,ln);
    if(cnt!--1)
    {
        int i,n;

        n=-1;
        for(i=0;i<cnt;i++)
        {
            if(p_tab[i].ptr==ptr)
                fprintf(logfile," ptr was previous allocated\n");
            else if(p_tab[i].ptr==NULL) n=i;
        }
        if(n==-1) n=cnt++;
        p_tab[n].ptr=ptr;
        p_tab[n].size=size;
        amount+=size;
        if(cnt>=PTAB_SIZE)
        {
            cnt=-1;
            fprintf(logfile,"Malloc table overflow\n");
        }
    }
    fflush(logfile);
}

```

joined. "NEVER JOINED!?" - That's right, I never joined. So what can I do about it now?

What Now?

Now, is the time to get involved. More, to STAY involved, with a group that has as much potential to promote our community as Microware itself.

Join now. I am. Send those checks to Carl Boll. If you're already a member, send your 'refund' money to Carl. Yes, send that cold hard cash, for without it your Users Group will not be. But remember, always remember, that money is not enough. It takes more than money to make an organization work. It takes more than money to make things happen.

It takes will. It takes knowledge. It takes determination. It takes Vision. Now is the time to contribute more than your money. Give of your selves to create a group as great as all of us. Communicate your ideas and comment on those advanced by others. Give your time, and your knowledge. Make a place in your life for a User's Group that is an active force for progress in the OS9 community.

What's In It for me??

Good question! What's in it for me? Well, I can tell you what I want to see happen to OS9. Me - the CoCo User. Me - the MM/1 owner. Me - the OS9 computer retailer. Me - the OS9 computer manufacturer. The CoCo user and MM/1 owner in me wants to see applications and articles about them. In fact, I want to see source code! I want to see general computing stuff that I can use for personal productivity. The retailer in me wants to see applications that I can use to prepare advertisements and business letters as easily as those MS-Dos using downs can, and games that sell big! The manufacturer in me wants an easy way to locate standards, engineers and programmers to use those standards, and custom-

ers to buy the computers and components that I build. Do I think that a User's Group can do all that? No. But it could help.

Long Term Goals.

We need some. Plain and simple. Those should be set within 6 months. What should they include? Simple. The Users Group should set up programs for several markets. The priority should be acquiring people with experience in each market to manage the programs, as well as active participants. The programs should center around personal users, industrial users, applications programmers, and systems programmers.

The specific aspects of each program should be the result of input from members drawn from the target group, and carefully crafted to yield maximum result from minimum input of cash and effort. The Users Group should provide organizational structure and operational guidance to those programs, but provide actual labor only where it can be easily done within the time available to the officers or a single volunteer.

For example, the maintenance of a programmers registry could be done by one volunteer, reporting to an officer. The porting of the GNUish library could be done quickly by a group of volunteer programmers acting as a work group directed by an officer. Programming standards could be set by a committee drawn from throughout the community, chaired by a member, and simply report to an officer. These, and other projects, benefit the whole OS9 Community. When OS9, in whatever form, prospers, we, the citizens of the OS9 community, prosper. Let us consider these and other ideas, in the coming months. Let us for once, put some effort into building an organized future for OS9. Write to the Users Group, share your ideas. Offer your expertise. Work with the OS9 Users Group for a better tomorrow.

-David M. Graham



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!

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



**FAT CAT
Publications
Bookshelf**

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

when things go wrong.

Please note that you may have to do some customization for your own projects. For example, I use my own terminate() function rather than exit() because my programs usually fiddle with the terminal path descriptors and terminate() restores things properly...in a simpler program you might just call exit(). You might also want to exit() if any error conditions are encountered.

Memcheck is not complete. However, until I get another hard-to-track-down bug I'll probably just leave it alone. Things to add:

- support for realloc(). If you do this you'll have to remember that realloc() can expand an existing memory block, or allocate an entirely new area of memory.

- adding boundary sentinels. This would involve allocating some extra memory on each call, placing a know value at the start and end of each block, and then checking the blocks when free(ed) to ensure the sentinels have not been corrupted. Remember that on the 68000 malloc() returns a pointer which is aligned for any variable, so the start sentinel should probably be a 32 bit integer.

This was written for the C compiler which comes with OSK. This compiler, as all ANSI compliant C compilers, automatically defines the macros `__FILE__` and `__LINE__`. However, you could probably make some minor changes and use it on Level II. First, at start of each file in your program you'd have to something like:

```
#ifdef MEMCHECK
#define __FILE__ thisfile.h
#endif
```

and then at the start of each function:

```
#ifdef MEMCHECK
#define __FUNC__ thisfunction
#endif
```

and finally, change the `__LINE__` references in "memcheck.h" to `__FUNC__`.

Heck, if you are real clever you could probably write a macro which is defined in "memcheck.h" to do it all for you. Let us know!

If you are having erratic bugs in your program, don't be stupid like I was and try to find the bug by testing and looking at code listings. Have the computer do some of the work for you!

I've really enjoyed all the cards and letters from you. But don't stop now. Send in those ideas and comments. I can be reached at PO Box 355, Porthill, ID 83853 or PO Box 57, Wynndel, BC, Canada V0B 2N0 or Compuserve 76510,2203.

LISTING 1

```
/* memcheck.c */
#ifdef MEMCHECK
/*
```

* 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

This file replaces some common memory allocation routines with special debug versions. To include the new routines you must have memcheck.h included in your sources and have MEMCHECK defined.

FINDING Malloc() PROBLEMS

By Bob van der Poel

It happened again the other night: I got a message from a distributor advising me of a rare bug in one of my programs. No, he couldn't reproduce it...but at times the program generated a **ERROR #102** on his OS9/68040 system. I spent hours looking at my source (in the area I figured the bug *had* to be). No luck. In desperation, I dusted off a project I'd started over a year ago—and I traced down the bug in less than five minutes. Now, I only wish I'd finished off "memcheck" a lot sooner.

Problems with dynamic memory allocation are pretty common in C programs. What happens is that a program uses malloc() to get some memory and then later it either:

1. modifies the pointer malloc() returned and then free()s memory which malloc() never gave out, or
2. returns the memory via free(), but continues to use it, or
3. uses more memory than was requested in the malloc() call, or
4. uses a pointer which malloc() never did issue.

Careful programming can certainly take care of all these problems. But, in a large, complex program mistakes can creep in. In my own programs I've found that #1 and #2

are the most common. The technique I use in my memcheck routines are nothing new. I've seen lots of references in books and magazine articles with similar projects. Of course, mine is a bit different. Plus, I can vouch that it really does catch bugs.

The theory behind the routines is pretty simple. Replace the malloc(), free() and other dynamic memory calls in your program with a set of routines which does more monitoring. Your replacement routines will then tell you what, if anything, is going wrong.

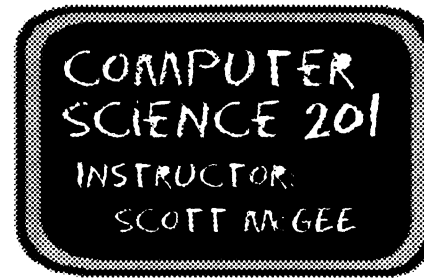
Using the C preprocessor, it is pretty easy. You write replacement routines and use a #define to redefine the standard functions. I've done this in the file "memcheck.h". This file should be #included in all the source files in your project. If you wish memory checking enabled just change your makefile to define MEMCHECK. Here's an abbreviated makefile I use:

```
MEMCHECK = -DMEMCHECK
... more macro defs
CFLAGS = -qs -t=$(TMPDIR) -x $(MEMCHECK)
... more make stuff
```

If I want to disable the memcheck routines I just put a "#" after the "=" (MEMCHECK = #-DMEMCHECK). Now the marco has a null definition.

I also include "memcheck.c" in my list of source files for the project. Again, using the same makefile this module gets compiled to absolutely nothing if MEMCHECK is not defined...or it contains the check functions if it is. Important: Do not #include memcheck.h in memcheck.c!

Hopefully, the comments in the listing will give you enough details to use these functions. The memcheck_xx() routines replace their more common relatives. A table is generated which contains all the allocations. This table is checked by memcheck_free() when any memory is released. The memory log contains a listing of all the allocations and free()s with a notation

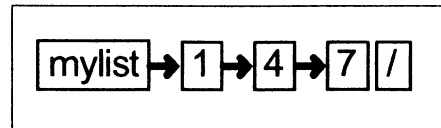


LESSON 2: LINKED LISTS CONTINUED INSERTION AND DELETION

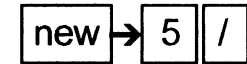
Last time, we discussed linked lists. We covered creation and deletion of linked lists, and talked about traversal of them. We didn't have time to cover two very important subjects, insertion and deletion of a single node to/from within the list.

Both of these operations are made more complex that they first appear, because if the node is inserted/deleted within the list, several links must be updated carefully to maintain the list.

First, let's discuss insertion. A new node with the needed value is constructed much as we discussed last time. I won't bother with code for that. Next, a process much like our traversal is used, but with the addition of a test, to find the location for the new node. As an example, let's assume that we have created the list with code similar to that used last time, but increment by three instead of one each time. Therefore, the list will look like this:



In addition, we have created our new node, with the value of 5, that we wish to insert. This item looks like this:



What we want to do is to keep the list in order, so that the new node must be inserted between the four and the seven. If we change the while loop in our traversal routine to read

```
while(node->data < new->data){
    parent = node;
    node = node->next;
}
```

The variable parent (of type list) points to the last node looked at, while the variable node is updated to point to the next node to examine. Once this code terminates, parent will be pointing to the node containing five, and node will be pointing the list element containing seven. To insert new in the list at the desired location, we must make the next pointer in parent point to the new node, while the new node's next pointer must point at node. The following code should do the trick.

```
new->next = node;
parent->next = new;
```

While this code seems extremely simple, there are several things to pay close attention to. First, the use of parent allows us to get the nodes on both sides of the new node in a single pass. Second, having different variables for all three nodes being manipulated lets us change values without having to be careful of ordering problems. If, for instance, we had only identified the node after which we wish to place the new node (parent in this case) but not the node after that (node), we could easily use the following code to insert

the new node:

```
new->next = parent->next;
parent->next = new;
```

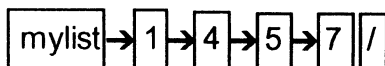
but if we accidentally coded these two lines in the opposite order:

```
parent->next = new;
new->next = parent->next;
```

we would have lost the pointer to the remainder of the list, and then set the next pointer of the last node (new) to point to itself. This is VERY easy to do, and care must be used to avoid it. Since we had a pointer to all three nodes in the first example, we can easily reverse the two lines without losing anything.

A very similar procedure is used to delete a node. In this case, the traversal loop will identify the node to be deleted and its parent, but we will not use an extra variable for the node after the node to be deleted, but will have to be careful of the order we do things in.

Lets assume that our list is the one resulting from our insertion process. It will look like this:



If we wish to delete the node containing four (ok, we have a variable called target of type int, with the value of four, that indicates the value of the node we wish to delete), we would do something like this:

```
while(node->data < value){
  parent = node;
  node = node->next;
}
parent->next = node->next;
free(node);
```

Again, care must be used in the order we do things. If we reverse the order of the last two lines, and free node too soon, we no longer have a valid pointer to the remainder of the list. In fact, if some other process allocates the newly freed memory before we do the "parent->next = node->next" line, we may actually end up pointing into some other area of memory used for something else entirely. Therefore, we must be sure to update things in the right order.

Now, we have taken care to correctly delete and insert nodes from within a list, but will this same code work if the node is either the first or the last node in a list? What if there is only one node in the list? How about an empty list? All four of these conditions must be examined to be sure our code is usable. If not, we will have to have separate code for any cases for which it fails, and code to test for that case. I'd rather make sure it works for all cases!

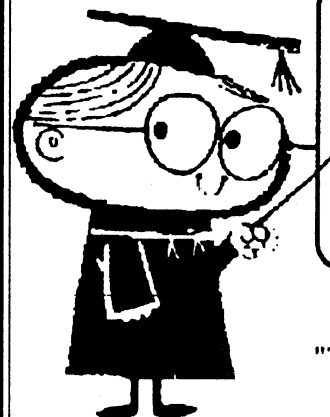
First, will they both work if the node is at the end of the list? In fact, yes, they both do. In the insert code, parent points to the last node, and node points to NULL. After new is added after parent, the NULL is copied to new->next. This is ok. For delete, node points to the node to be deleted (the last node in the list) and parent to the node before it. We will end up copying the NULL from node->next to parent->next and all is fine.

How about the first node in the list? Well, since the interior of the loop is never executed, parent is never set. Uh oh! Well, what can we do to fix it?

One thing to do is to handle this as a special case, but like I said, I would rather not. The other option is to use a pointer to a llist called node_address like we did to create our list originally. This way, node_address is initialized to the address of list before the loop, and all references to node would be replaced with references to (*node_address).

Teach Your Computer New Tricks

Use *InfoXpress*→



✓ SAVES TIME

✓ SAVES MONEY

✓ LESS AGGRAVATION

"Today's lesson: Don't be a slave to your machine."

- * Automatically logon to Compuserve and Delphi
- * Download your waiting electronic mail and forum messages
- * Review your messages and reply offline
- * Upload your replies at the next online session

Available from: Dirt Cheap Computer Stuff Company
1368 Old Highway 50 East
Union, Missouri 63084
314/583-1168

\$49.95

CoCo Version

\$69.95

OSK Version

```

xcount++;
if(board[Row][Col] == 'O')
    ocount++;
}

if(xcount == 3)
    End = 1;
else if(ocount == 3)
    End = 2;
}

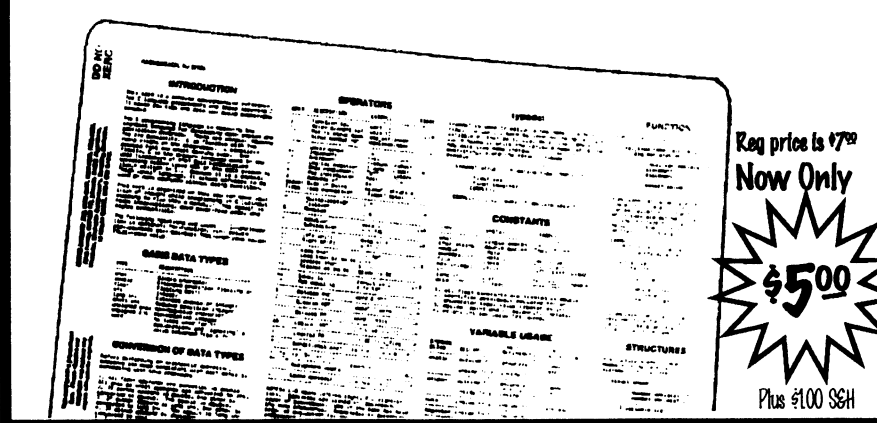
/* This checks for a right to left diagonal win. */
xcount=0;
ocount=0;
for(Row = 0, Col = 2, Row < 3 && Col > -1; Row++, Col--
{
    if(board[Row][Col] == 'X')
        xcount++;
    else if(board[Row][Col] == 'O')
        ocount++;
}

if(xcount == 3)
    End = 1;
else if(ocount == 3)
    End = 2;
return(End);
    
```

Due to limited space this month, "socremod.c" will appear in next month's issue. -Editor

AniMajik Productions
 4650 Cahuenga Blvd., Ste #7
 Toluca Lake, CA 91602
 (818) 761-4135

The C Language Instant Reference Micro Chart is now on Sale..!
 The MICRO CHART measures 8 1/2" x 11" and is 100% plastic for durability...and spill-proof!
 The perfect companion for your computer desk. This is the concise K&R C Reference that puts most
 everything right at your finger tips. List commonly used functions, CMP Line ARGs, Printf and Scanf syntax,
 TypeDef, Constants, Variables, Operator and Statement summaries.



Reg price is \$7.99
Now Only \$5.00
 Plus \$1.00 S&H

The loop for insert becomes:

```

node_address = &list;
while((*node_address)->data < new->data){
    parent = (*node_address);
    node = &((*node_address)->next);
}
    
```

[Note: I like to fully parenthesize expressions, the compiler removes any redundant parentheses, but cannot insert missing ones!]

This change works identically for middle and last cases, but correctly handles the case of the node being first. A similar change will fix delete.

Now, how about when there is only one node in the list. This really combines the first and last cases, and in this code (with the fix mentioned above), our code works here.

Finally, what if the list is empty? For insert, a change to parent similar to the one made to node would fix it. In this case, however, we are dealing with a degenerate case, and I would almost rather just add a special case to handle it due to the complexity of the other approach. To do so, add the following code before the rest:

```

if(list = NULL)
    list = new;
else{
    
```

and be sure to terminate the else (with a ') after the remaining code.

Delete is also a degenerate case, and can also be handled with a special case. I'll leave the details as a homework assignment.

OK, now we have one more thing to consider. What if a list may contain two or more items with the same value? Does this change anything? If we don't care what order identical items appear in the list, I don't think so. The insertion code will work as is. It will insert a new item BEFORE other identical items.

If we wanted it after, we would have to adjust the comparison accordingly. To delete, our code is also OK. It will delete the first of any group of identical items. Again, if we want the last, just adjust the comparison. Another thing to consider here, is what if these nodes are more complex? Lets say that they contain other fields. In this case, we may want to modify our comparison to be able to compare other fields if the data field is identical.

One last thing to consider. What do we do if we are asked to delete an item that is not in the list? Our previous code blindly assumed that if we passed everything that was less than the desired node, the next node was the one we wanted to delete. Obviously, this leads to undesirable results if the target value did not exist. What we have to do is to add a second test somewhere in the code that will assure that the delete operation is only performed (after the search loop terminates) if we really have the right node. It must just terminate the delete operation if not. We may, or may not, wish to generate some sort of error message when this happens.

As part of your homework assignment, in addition to the code to handle delete from an empty list, you are to rewrite the delete code to correctly handle the target node not being in the list. I expect everyone to have a correct, fully functional delete routine by the next time we meet.

Class dismissed!
 -Scott McGee

email: _____
 Internet: smcgee@cymru.UUCP
 or smcgee@cymru@uunet.uu.net
 UUCP: uunet.uu.net/cymru!smcgee
 STGNet: smcgee@os9er
 snail mail (US Mail):
 Scott McGee
 c/o The OS-9 Underground
 4650 Cahuenga Blvd., Ste #7
 Toluca Lake, CA 91602

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

Technical Assistance & Inquiries
(216) 758-7694

JWT Enterprises
5755 Lockwood Blvd.
Youngstown, OH 44512



Copyright: 1991

OS-9 is a trademark of Microware 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.

```

else if(End == 2)
  print("\nHey, whimpering simp! Wanna play again? (y/n)");
else if(End == 3)
{
  print("\nLook. One of us is smarter—I think it's me. Want to");
  print("find out? (y/n)");
}

scanf("%c", &Choice);
if(Choice == 'y')
{
  Goofy = 1;
  printf("\nSmart move...\n");
}
else
{
  Goofy = 0;
  printf("\nThat was cute.. Bye\n");
}

return(Goofy);
}

Function Name: CompMove
Purpose & Operation:
Determines which square to place an O, using the score
algorithm provided by Kent Woodriddle. Stores the O
in the board array variable which represents the logical
square to pick in order for the computer to win.

Input Variables : none
Functions Called : score
*/

void CompMove(void)
{
  int Scor=0;
  int MaxRow=0;
  int MaxCol=0;
  int Max=0;

  int Row;
  int Col;

  for(Row = 0; Row < 3; Row++)
    for(Col = 0; Col < 3; Col++)
    {
      Scor = score(board, Row, Col);
      if((board[Row][Col] == 'X' & board[Row][Col] == 'O'))
        Scor = 0;
      if(Scor > Max)
      {
        MaxRow = Row;
        MaxCol = Col;
        Max = Scor;
      }
    }
}

board[MaxRow][MaxCol] = 'O';
printf("\nMy move is %d %d.\n", MaxRow+1,
MaxCol+1);
}

Function Name: EndCheck
Purpose & Operation:
Checks for a win for either the user or the computer, and
also checks for a tie. Returns a flag to the main function
depending on whether there is a win or a tie.

Input Variables : none
Functions Called : none

int EndCheck(void)
{
  int Row=0;
  int Col=0;
  int End=3;
  int xcount=0;
  int ocount=0;

  /* This procedure checks for a Cat's Game */
  for(Row = 0; Row < 3; Row++)
    for(Col = 0; Col < 3; Col++)
      if(board[Row][Col] == 'O')
        End=0;
  /* This checks for a horizontal win. */
  for(Col = 0; Col < 3; Col++)
  {
    xcount=0;
    ocount=0;
    for(Row = 0; Row < 3; Row++)
      if(board[Row][Col] == 'X')
        xcount++;
      if(board[Row][Col] == 'O')
        ocount++;
  }
  if(xcount == 3)
    End = 1;
  else if(ocount == 3)
    End = 2;
}

/* This checks for a vertical win. */
for(Row = 0; Row < 3; Row++)
{
  xcount=0;
  ocount=0;
  for(Col = 0; Col < 3; Col++)
  {
    if(board[Row][Col] == 'X')

```

```

DisplayBoard();
if (End == 1)
    printf("\nCongratulations! You win!\n\n");
else if (End == 2)
    printf("\nSucker! I win!\n\n");
else if (End == 3)
    printf("\nCat's game! A tie—I'll get you next time!\n\n");
Repeat = Again();
return(0);

```

Function Name : DisplayBoard

Purpose & Operation :
Displays the tic-tac-toe board.

Input Variables : none

Functions Called : none

```

void DisplayBoard(void)
{
    printf("\n");
    printf(" 1 2 3\n");
    printf("%1.1c | %1.1c | %1.1c\n", board[0][0],
        board[0][1], board[0][2]);
    printf(" -+-\n");
    printf("%2.1c | %2.1c | %2.1c\n", board[1][0],
        board[1][1], board[1][2]);
    printf(" -+-\n");
    printf("%3.1c | %3.1c | %3.1c\n", board[2][0],
        board[2][1], board[2][2]);
    printf("\n");
}

```

Function Name : UserMove

Purpose & Operation :
Inputs the user's selection for a move and places an X in the board array, corresponding to the square selected.

Input Variables : none

Functions Called : none

```

void UserMove(void)

```

```

{
    int Row;
    int Col;

    Row = 4;
    Col = 4;

    while ( Row > 3 && Col > 3 )
    {
        printf("Your move? ");

```

```

scanf("%d %d", &Row, &Col);
if ( (Row > 3) | (Col > 3) )
    printf("\nYou think I'm stupid or something?\n\n");
if ( (board[Row-1][Col-1] == 'X') |
    (board[Row-1][Col-1] == 'O') )
{
    printf("\nNice try...you won't beat me by cheating, that's for");
    printf("\ndamn sure!\n\n");
    Row = 4;
    Col = 4;
}
board[Row-1][Col-1] = 'X';

```

Function Name : ClearBoard

Purpose & Operation :
Places # space in each member of the board array that represents the squares of the tic-tac-toe board, clearing all the squares.

Input Variables : none
Functions Called : none

```

void ClearBoard(void)

```

```

{
    int Row;
    int Col;

    for(Row = 0; Row < 3; Row++)
        for(Col = 0; Col < 3; Col++)
            board[Row][Col] = '#';
}

```

Function Name : Again

Purpose & Operation :
Lets the user input whether he wants to continue or not, and displays whimsical, and utterly ingenious sayings depending on that choice.

Input Variables : none
Functions Called : none

```

int Again(void)

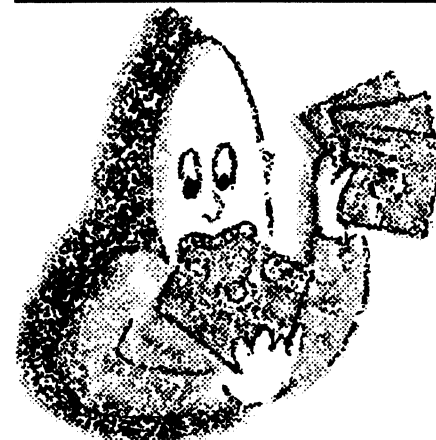
```

```

{
    int Goofy;
    char Choice = 'q';

    if (End == 1)
    {
        printf("\nOh, Tic-Tac-Toe Master, would you like to ");
        printf("play again? (y/n) ");

```



MASS QUANTITIES

MAKING BACKUP DISKS

BY ERIC LEVINSON

I have been using FBU (Floppy Backup) by Frank Hogg Labs since 1987. It is a backup/restore utility that will copy the file structure from your hard disk onto formatted disks. This program is very similar to AR which will archive many files including full pathlists into one file. What AR will not do is span many disks, unless you have a program that will take the archive stream and break it up between disks. Prior to backing up your hard disk you must format disks in mass quantities (conehead??).

The utility in this article is an auto formatter. This program will ask you key questions, and then format disks in a timely fashion. The only thing you have to do is enter the drive descriptor (on my system it is /D1). Next, the program will ask you for a volume name. This name could be the name of your backup. I use something like "MondaysBackup" or "TuesdaysBackup". It is a good idea to have more than one set

of backup disks, and in some cases, you should have one set for each day in case of a corrupted set. The next question asks you for which backup number disk you want to start the auto formatter at. You type in some number. If this is your first disk, just enter 1 and press enter. Finally the program tells you it will format the disk when you answer the next question. You can either just press enter or X. If you press X, the formatter will exit. If you press any other key formatting will take place immediately.

This program will automatically label diskettes in succession. If there was a problem with making disks, the auto formatter will allow you to continue from where you left off. Remember, there will be no warning coming from the format command. It starts automatically.

There are basically two types of backup methods. Both of which require a stream of data. The types are file-by-file or track-by-track. The file-by-file backups are the most widely used in larger systems, because areas of the hard disk that are not used are not even read. Also, the file-by-file backup, when restored to a formatted hard disk is completely fragment free, as if the hard disk was defragmented. In the track-by-track method, the software "reads" the hard disk like a record, starting at the beginning and running until the end. Any problems with the media or file structure will be replicated in the backups.

Both types of backup methods have some sort of compression algorithm which cuts down on the amount of backup disks needed. The file-by-file method would allow a particular group of files to be restored instead of the whole set.

In short, this program will allow you to make a sandwich (or 100) while your Color Computer does it's thing. You won't have to do anything except swap disks, and press the enter key. **Autofmt** will do the rest! Happy Backup!

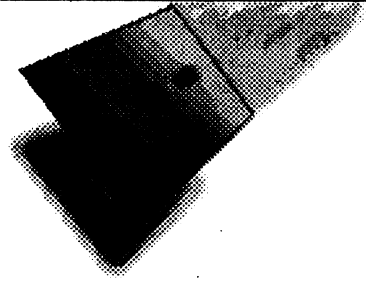
-Eric Levinson

(Listing continues next page)

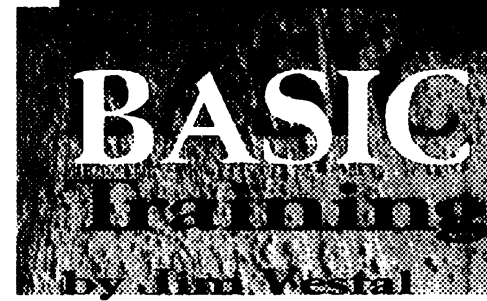
```

PROCEDURE autofmt
REM AUTOFMT (C) 1987 Eric Levinson
REM All Rights Reserved
REM
DIM diskname:STRING[32]
DIM drivename:STRING[3]
DIM key:STRING[1]
DIM word:STRING
PRINT "This will format disks over and over again until you press X"
INPUT "Drive ",drivename
INPUT "Volume name: ",word
PRINT "Start at "; word; " #"
INPUT x
LOOP
    diskname:=word+"#" +STR$(x)
    PRINT CHR$(7); "Warning: Will format after answering the next question"
    PRINT "Enter X to end"
    INPUT key
    EXITIF key="X" OR key="x" THEN ENDEXIT
    PRINT "Formatting "; drivename; " as "; diskname
    SHELL "Format "+drivename+" r "+CHR$(34)+diskname+CHR$(34)
    x:=x+1
ENDLOOP
PRINT "Job terminated"

```



(EOF)



ANSI C is currently one of the hottest topics in the OS-9 world thanks to GNU C for OS-9 68k and Microware's Ultra C for OS-9 68k and OS-9000. Up until very recently OS-9 Level 2 users were left behind without a ANSI compatible C language. CoCo users now have a new freeware tool called ANSIFRONT written by Vaughn Cato. ANSI front allows CoCo OS-9 C programmers the ability to run standard C code, without having to worry about the incompatible VOID statements and the ANSI prototyping. This month I am submitting an ANSI C program. It's a simple tic tac toe game, written by my friend who is a computer science student at CSU Chico. It shows how to use the VOID statement and how ANSI prototyping works. To compile this source with the CoCo OS-9 C be sure to install cc 2.5 and the ansifront module.

Available on most bulletin boards or from 09-Online systems. OSk systems require Ultra C or GNU C, unless you to manually convert the VOIDs and the prototyping. Enjoy!

-Jim Vestal

file: ttt.c

Standard ANSI C program subitted to the OS-9 Underground-Basic Training

Program Name : Tic-Tac-Toe
 Programmer : Greg Chance
 (Score function by Kent Woodridge)
 Date : April 27, 1993

*/

/* Functional Description :

The classic, but simplistic, game of Tic-Tac-Toe. The rules are simple. You try to win by placing 3 X's in a row horizontally, vertically, or diagonally. You play against the computer.

/* Operational Description :

The program formats the normal Tic-Tac-Toe board with numbers along the sides of the board to ease program input. The computer's moves are determined by a routine that was supplied by Kent Woodridge. Each square has a calculated score. The square with the highest score is the computer's logical choice for a move. As with all tic-tac-toe games, winning is almost impossible.

/* Input File(s) : none Output File(s) : none
 Global Variables : board = the array for all the squares in the tic-tac-toe board. End = the user's choice as to whether he wants to play again.

```

#include <stdio.h>
#include "scoremod.c"
void DisplayBoard(void);
void UserMove(void);
void CompMove(void);
void ClearBoard(void);

```

```

int EndCheck(void);
int Again(void);

```

```

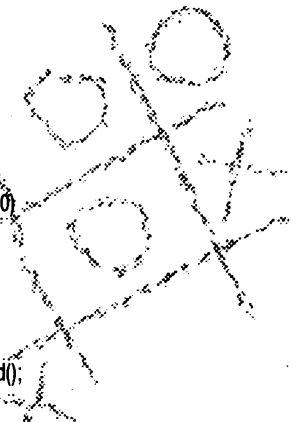
int End;
char board[3][3];
main()
{

```

```

    int Repeat= 1;
    while(Repeat != 0)
    {
        ClearBoard();
        End = 0;
        while(End < 1)
        {
            DisplayBoard();
            UserMove();
            End = EndCheck();
            if(End < 1)
            {
                DisplayBoard();
                CompMove();
                End = EndCheck();
            }
        }
    }
}

```



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!)

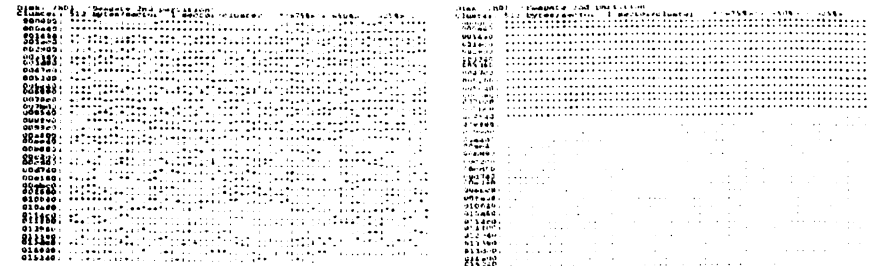
Subscriptions are **\$23/year for 8 issues** (every 6 weeks), or **\$12 for a 4 issue trial subscription** (\$30/\$16 for Canada, \$33/\$17 overseas). A disk service, "microdisk", is **\$40/year or \$6 per issue** (\$44/\$7 Canada, \$54/\$8 overseas). **First issue will be delivered in August... DON'T 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 (48 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
- LSrcDbg - 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.



C Software Engineering

by Leonard Cassady

In the last installment, we discussed the differences and additions of preprocessor implementation according to the ANSI C Standard. Some preprocessor features are distantly related to compiler lexical translation. While the rules for preprocessor directives appear to be the same as those for the compiler in some cases, compiler translation and preprocessor translation are distinctly different and the two should not be confused.

LEXICAL ANALYSIS

The characters in a C source program are collected according to the rules of the particular implementation into lexical tokens. The compiler forms the longest possible token by parsing in a left-to-right order. Divided into five classes, (operators, constants, separators, identifiers or names, and reserved words), they make up the execution set for the C language.

Names

Each C implementation is free to set their own limit to the length of declaration names. The ANSI C Standard requires compilers to support internal names of at least thirtyone characters and external names of at least six characters. Older compilers usually supported up to eight characters for

internal and external declarations.

Declarations may contain letters, numbers, the underscore character, or any combination of these, but MUST start with a letter or underscore.

Declarations in C are difficult to describe because of the abstract properties (scope and extent), and type casting. When the same identifier is associated with more than one program declaration, the name is 'overloaded', and program context determines which declaration is in effect.

There are five 'overloading classes', or 'names spaces', in ANSI C, and they differ slightly from Traditional C.

Traditional C restricts the use of member, or component, names to one type class unless their offsets in a structure are identical. This

restriction has been corrected in the more recent pre-ANSI implementations as well as ANSI C. When a name is overloaded with multiple declarations of different overloading classes, each declaration has its own scope. (See Fig. 1)

As an example, the same identifier may be used both as a variable and a structure tag and the the variable may be redefined without altering the tag association, or scope.

Prototyping

The scope of a declaration is the region of the C program code over which the declaration is active. The ANSI C Standard introduced a new declaration scope called 'prototyping'.

Introduced to the C Language by Bjarne Stroustrup of AT&T, and later adopted by the ANSI committee as part of the ANSI C Standard, function prototypes allow function allusions to include data type information about arguments. The function allusion syntax is the same as traditional C implementation, except that you may enter data types for each function argument.

Example:

```
"\xabc" /* This contains one character */
"\xab" "c" /* This contains two characters */
```

Some Traditional C implementations permit only up to a fixed number of hexadecimal digits.

As a general rule, is it better to hide escape code sequences in macro definitions. This practice also takes advantage of macro expansion and values only need be changed in one place.


This is the second installment of several in a series on the additions and differences introduced by the ANSI C Standard.

Any comments, suggestions are welcomed and should be sent to:

Software Engineering Assoc.
Standard ANSI C Library
6530 Independence Ave Ste # 168
Canoga Park, CA 91303

or emailed:
"ZOGster@AOL.com"
"zog!leonard@abode.ttank.com"





Type'writer: n. archaic machine for writing by the impression of type letters; one of the reasons home computers were created.

Write-right: n. a "real" word processor for the MM/1 by Joel Mathew Hegberg a complete word processor with "what you see is what you get" display of bold, italics, underline, and colors as well as a complete mouse-driven, cut-n-paste, on-screen text ruler, definable margins and tabs, insert/overstrike modes, text formatting, word-wrap and definable printer codes.

much more: n. what else write-right has to offer for your 100% K-Windows compatible OSK computer.

Trash the typewriter. Purchase Write-Right today for \$59.95 (plus \$2.50 shipping and handling) direct from Sub-Etha Software. For more information download the demo version ("WR_DEMO.LZH" found on GENie or Delphi) or call (815) 748-6638.

Send to:

SUB-ETHA SOFTWARE
P.O. Box 152442 - Lufkin, Texas 75915

Trigraph	Replaces
??=	# (pound sign)
??-	~ (tilde)
??!	(vertical bar)
??'	^ (caret)
??/	\ (backslash)
??<	{ (left brace)
??>	} (right brace)
??([(left bracket)
??)] (right bracket)

A trigraph-like character sequence could be translated as a trigraph by the compiler. In order to prevent this, a new character escape, '\?', is available for use in string constants.

Traditional string	Trigraph form	Result
"What??!"	"What?\?!"	What??!
"Backslash: \\"	"Backslash: ??\??\"	Backslash: \

Escape Sequences

Special characters, or character escape codes, are represented in a fashion that is independent of the computers implemented character set. The K&R Standard provides seven such character escape sequences, and the ANSI C Standard includes an additional five to make a total of eleven escape code sequences.

The backslash character, by itself, is translated as a continuation character by the preprocessor and is used to join two separate lines into one logical unit. When it is followed by a end-of-line marker or a whitespace, the compiler considers it the same as a comment and it is ignored. When followed by a any other character or numeric, it is considered the escape-character

of the 'escape-character escape-code' sequence. The escape codes come in two varieties, 'character escapes', which can be special characters used for formatting, and 'numeric escapes' which allow a character to be specified by its numeric encoding.

The ANSI C Standard introduced two new character escape codes, '\a' and '\v', and one numeric escape code, '\x', to those already defined in the K&R Standard.

escape-character: backslash, (\).
escape-sequence: escape-character character-code escape-character hex-escape-code escape-character octal-escape-code
character-escape-code: (one of the following) a b f n r t v \ " ' \a ?
octal-escape-code: octal-digit octal-digit octal-digit octal-digit octal-digit octal-digit
hex-escape-code: lowercase 'x'. hex-escape-code hex-digit

In ANSI C, the lowercase letters are reserved for future language features, and the uppercase letters are reserved for implementation defined extensions.

The programmer needs to wary of using numeric escapes, as they may depend on character encodings and might be non-portable. Also, the correct syntax is very particular. Octal escape codes terminate when three octal digits have been used or when the first non-octal character is encountered.

Example:

```
'\101' /* This contains one character */
'\1011' /* This contains two characters */
```

The second, multi-byte example is illegal and the compiler should issue an error or warning. Hexadecimal sequences suffer from a similar problem. Since they can be of any length, multi-byte sequences should be broken into pieces.

Class	identifiers
preprocessor macro names	Preprocessing occurs before compilation and macro names are independent of any others used in the C program.
statement labels	Named statement labels are always followed by a ':', (A 'case' label is not a statement label), and is always immediately follow the reserved word, 'goto'.
struct, enum, and union tags	Tags are always immediately follow the reserved words, 'struct', 'enum', and 'union'.
component or member names	Definitions of component or member names always occur within structure or union type identifiers. Use of component or member names always immediately follow the selection operators, '.' and '>'. The same name may be used in any number of structures or unions at same time.
other names	Variable, function, typedef, enumeration constants and all other declarations fall into this category.

Fig 1

Example:

```
extern void foo( a, b, ptr);
or
extern void foo( int a, float b, char *ptr); /* prototyping */
```

body may contain 'blocks' or compound statements. Each block may contain any number of 'inner declarations'. 'Extent' is the period of time these objects have memory storage allocated at run-time. Typedefs, unions, and structures, do not have extent as no storage for them is allocated at run-time.

The main benefits prototypes offer are:

- Compiler type checking of the actual function arguments to the formal arguments specified in the function allusion.
- Prototyping turns OFF automatic argument conversions. Algorithm speed can be significantly increased when using short integers and floating point precision.

There are other, more subtle effects prototypes offer. I'll defer the more detailed discussions to the numerous volumes on ANSI C programming that are available in book stores.

Extent (duration)

A C source file, sometimes known as a 'translation unit', consists of 'top-level' declarations which include functions, variables, and other objects. Each function has parameter declarations and a body. The

- Static extent - All functions, or variables and objects declared in top-level declarations. Memory allocation exists until program termination.
- Local extent - Storage is created upon entry to a function or block and destroyed upon exit. If a variable with local extent has an initializer, it is initialized each time it is created.
- Dynamic extent - Objects created or destroyed at the programmer's discretion. Dynamic objects MUST be created through the use of the 'malloc' family of library routines.

Formal parameters have local extent, and variables declared at the beginning of program blocks may have local extent, depending on their declaration. Variables with local extent are also known as 'automatic variables'.

Continuation Character

The backslash followed by a whitespace appearing in a string literal, or string constant, is used to span the string over one or more lines.

The backslash and end-of-line marker are removed to create a longer, logical source line. The splicing of source lines occurs before preprocessing and before the lexical analysis of the C program.

Prior to the ANSI C Standard, the continuation character could only be used in string literals. While NOT specified in the ANSIC Standard, some ANSIC implementations allow the splitting of variable names and tokens between source lines.

Example:

```
if(argv[argc] != index[i]) ; e\
se color_index[i] == index[i];
```

Most C implementations restrict the maximum length of source lines both and after line splicing. ANSI C requires logical source lines to be of at least 509 characters in length.

String Concatenation

String concatenation can be used to break up long strings that would normally require the continuation character. The ANSI Standard states that two adjacent string literals will be concatenated into a single null-terminated string.

Example:

```
printf("Now is the time for all good men to\
/* Traditional C */
come to the aid of their country.\n");

printf("Now is the time for all good men to"
/* ANSI C */
"come to the aid of their country.\n");
```

Some implementations of both Traditional C and ANSI C incorrectly remove the leading whitespace characters from the concatenated string.

Trigraph Sequences

The C language character set that may be used in source code is much larger than those used by most other programming languages, and in some cases, are not always available on the computers' keyboard. The ANSI C Standard adopted a new format to overcome this deficiency.

A set of nine 'trigraphs' were introduced so that source code could be written using the ISO 646-1083 Invariant Code Set, (a 7-bit subset of the ASCII code set), that almost all keyboards support. Non-ANSI C compilers do not customarily support the trigraph sequences. They may introduce compile-time errors when porting ANSI C source code for use with a Traditional C compiler. The K&R Standard does not support the trigraph feature.

Trigraphs consist of two question marks followed by a third character. During the compiler translation stage, these trigraphs are converted into a single character. This translation occurs before lexical analysis, (tokenization), and before escape characters in string or character constant are recognized.

**Know someone who ISN'T
getting the Underground???**

**Tell 'em to Get Down...
...to the Underground!**

4650 Cahuenga Blvd., Ste #7
Toluca Lake, CA 91602
(818) 761-4135

Watch for details in your mailbox soon
that let YOU benefit from your referrals

Northern Xposure

'Quality Products from North of the Border'

BRAND NEW PRODUCT, SPECIAL PRICE!

Smash!

\$25.00

Breakout-style Arcade Game for OS-9 Level II

\$29.95 after Dec 25, 1993.

- 1 or 2 player mode
- 32 levels, can be edited
- 17 different block types
- multiple balls
- 320x192x 16 color graphics
- written by Alan DeKok, author of CC3Demo and Thexder:OS-9

*Many other fine products available
Write for a free catalogue*

- Prices are US funds and include S&H ●

Alan DeKok

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

Colin McKay