AUSTRALIAN OS9 NEWSLETTER

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-3236
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
SUPPORT	Brisbane OS9 Users Group	

Addresses for Correspondence

Editorial Material:

Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

Subscriptions & Library Requests:

Jean-Pierre Jacquet
27 Hampton Street
DURACK Qld 4077

Volume 5 March 1991 Number 2

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 5 Number 2

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries
TREASURER : Don Berrie
LIBRARIAN : Jean-Pierre Jacquet
SUPPORT : Brisbane OS9 Level 2 Users Group.

Well here again is our newsletter. This month we have included the second part of the unfrag programme from Bob van der Poel. We also have a submission from one of our members comparing the CoCo windowing system with Microsoft's Windows Version 3.0

We have also included some snippets from the Internet CoCo OS9 mailing list.

Quotes of the Month

"It is virtually impossible to do any serious multitasking with less than four megabytes of memory."

Bill Gates, CEO Microsoft

Writing programmes in machine code is like digging post holes with a teaspoon, slow and painstaking, but you have complete control over all of the dirt.

News Briefs

You can now use a serial mouse with CoCo Os9. Patches to CC3IO have been written by Bruce Isted in the US. We are trying to get hold of these patches for our PD Library

The MM/1 windowing system is very much like stock L-II. By the way, the windowing is running on a Atari ST also... with others to follow. (Running the same binaries on both the MM/1 and the Atari! Neat, eh?) Each machine has different capabilities and resolutions. The windowing should be flexible enough to handle all those without being the burden to programmers that most such windowing systems are.

All it is so far is exactly the same as Multi-Vue, with some added functionality. At this writing, the GUI works just like the CoCo's Multi-Vue. In fact, you can run your CoCo as a terminal off the MM/1 using graphics generated on one then displayed on the other. Mousing across machines is something else to be worked on.

The screen capabilities of the MM/1 will look fine. The x420 mode is interlaced, meaning you probably won't use it for text due to normal NTSC flicker (may not trouble us here in Australia), but it'll look great for pictures. The 720x modes simply overscan past the edges of the screen, which is intentional... although the monitor probably can be adjusted inward if wished.

All the screens on the MM/1 look fine on a CM-8 monitor, so the Commodore 1084S should look fine also. Some of the higher res. graphics screens flicker somewhat on the CM-8, but I'm told look better on the Commodore types.

List of Seagate SCSI Hard Drive prices versus price, with KBytes/buck (in US dollars) already calculated. It's clear that as you go up in size you get more for your money, up to 84 MB anyway.

MBytes	Price	KB/$	
20	259	77	not good value
30	279	107	RLL, steer clear
48	319	150	ST-157N, good value
64	349	183	RLL?
84	379	222	Best Buy
142	739	192	past the peak

All Seagate Hard drives with an N suffix have onboard controllers, and should simply plug straight into the SCSI interface on the MM/1.

Look for the name Realtime Multi Media as the Australian Agents for the MM/1. We are told maybe September or October!!

Maybe some of this material whets your appetite? We are always on the lookout for material for the newsletter. Maybe you can help? Until next time -

Cheers Gordon.

oooooooooo0000000000oooooooooo

Four Drives!
by Bob Devries.

Can you believe it? I have now four double sided floppy drives connected to my CoCo3! How can that be, I hear you say.... There are not four drive select lines, only three and the side select line. Well, that is true, and I had to do a bit of hardware hacking (my main source of enjoyment with the CoCo. After C programming of course!). Well, I'll tell you about how I did the mods. Let me first say that if you decide to try it for yourself, you're on your own! I will not be responsible for any damage.

OK, now the disk controller I have is a CRC-Disto Super controller II (with 4-in-1 fitted, but that's by-the-way). Now I would imagine that the mods should be able to be done to any disk controller, as long as ther is room to physically fit the extra chip needed. Besides that I have a standard 5 1/4 drive case with 2 80 track DS drives in it.

To make provision for connecting the extra drives (2 720K 3 1/2 floppies), I made an external drive connector on the back of the drive. I used a 23 pin DB connector and wired it along the lines of the Amiga A500 external drive connector. Here is the pinout I used.

DB23 pins	Standard pins
1. Ready (not used on COCO)	(34)
2. Read Data	(30)
3 - 7 Ground	(all odd pins)
8. Motor on	(16)
9. Drive select 2	(14)
10. Disk reset (not used on COCO)	
11. Disk change (not used on COCO)	
12. +5 Volt supply	
13. Side select	(32)
14. Write protect	(28)
15. Track 00	(26)
16. Write Gate	(24)
17. Write Data	(22)
18. Step	(20)
19. Direction	(18)
20. Drive select 3 (not currently connected - read on)	(6)
21. Drive select 1 (already used for 5 1/4 drives)	(12)
22. Index	(8)
23. +12 Volt supply	

OK so here's how I modified the controller. Firstly, you'll need a 74LS138 chip. Your local Dick Smith store will supply that, as well as three 10k ohm 1/4 watt or 1/8 watt resistors.

Here's the circuit:

```
                                  _  _
from controller drive sel 0      1 !  U  !16  connect to +5 volt
 "     "         "   "  1       2 !     !15  connect to drive sel 3 (pin 6)
 "     "         "   "  2       3 !     !14  not connected
       connect to ground          4 !     !13  connect to drive sel 1 (pin 12)
       connect to ground          5 !     !12  not connected
       connect to +5 volt         6 !     !11  connect to drive sel 2 (pin 14)
       not connected              7 !     !10  not connected
       connect to ground 0 volt   8 !     !9   connect to drive sel 0 (pin 10)
                                    ----
```

The three resistors must be connected one end to +5 volt, and the other ends to pins 1, 2 and 3 respectively.

This little circuit may be connected either in your disk drive case, or in the controller itself, where-ever there is sufficient space. I mounted mine inside the controller, placing the chip piggy-backed on top of another chip of similar size, with all the pins except pins 16, and 8 bent out sideways. I soldered these two pins to the chip underneath, making sure of two things: (1) that the chip underneath was another 74LSXXX chip or 74XX chip (e.g. 7416), and (2), that the piggy-backed chip was the same way 'round. Well, OK, I also checked that I did not blob the solder all over the other chips, but that's normal for any electronics practices.

To make the connection to the circuit, I cut the tracks just before the connection to the 34 way edge connector where the drive cable plugs in. Only three need to be cut, pins 10, 12, and 14. There should be no connection to pin 6, and the wire from pin 15 of the 74LS138 chip may be connected here. By the way, I used 28 guage 'Kynar' wire wrap wire, but any fairly thin gauge wire would do.

OK, so now how do you use these drives. Firstly let me say that DISK BASIC (shudder) cannot use the extra two drives, and will work normally without modification. OS9 needs to have a module patched. The module is CC3Disk. I strongly suggest that you use 'DED' to do the modification. Find the character combination that goes like this (in HEX) 01020440. This is the drive select table bit pattern. Change the byte 40 to 07. Don't forget to write the new module to disk, and verify to correct the CRC. You can do this from within DED. Please read the DED docs FIRST, and do it on a BACKUP of your normal system disk. Next, you'll need to add two new device descriptors /D2 and /D3 to your OS9Boot file. You will of course do this in the normal way (whatever that is). The /D3 descriptor you'll have to create yourself, or modify the one on the Boot/config disk named d3_35s.dd. Here is a dump of my descriptors to show you what they look like:

This is the one for /D2

```
    0 1 2 3 4 5 6 7  0 2 4 6
ADDR 8 9 A B C D E F  8 A C E
==== +-+-+-+-+-+-+-+- + + + +
0000 87CD00300021F181 .M.0.!q.
0008 D400230026FF07FF T.#.&...
0010 400F010203200300 @.... ..
0018 5002000012001203 P.......
0020 0844B25242C64343 .D2RBFCC
0028 33446973EB834376 3Disk.Cv
```

Here's the one for /D3

```
    0 1 2 3 4 5 6 7  0 2 4 6
ADDR 8 9 A B C D E F  8 A C E
==== +-+-+-+-+-+-+-+- + + + +
0000 87CD00300021F181 .M.0.!q.
0008 D400230026FF07FF T.#.&...
0010 400F010303200300 @.... ..
0018 5002000012001203 P.......
0020 0844B35242C64343 .D3RBFCC
0028 33446973EBE0187E 3Disk'..
```

Don't forget... I am using the CRC-Disto Super controller II in NO-Halt mode. The descriptors may be different from yours! CHECK THIS FIRST.

As I said earlier, I am using the four drive system for two extra 3 1/2 drives (720K ones) so that I can now read just about all format drives. I use SDisk3 for reading other format disks such as Atari OSK disks, and IBM PC/XT disks in both 5 1/4 and 3 1/2 format (using the PD PCDos programme). I wish there was some way of reading Amiga A500 disks... anyone know how?

Have fun hacking
Regards,
Rob Devries.

oooooooooo0000000000oooooooooo

CoCo-Link

CoCo-Link is an excellent magazine to help you with the RSDOS side of the Colour Computer. It is a bi-monthly magazine published by Mr. Robbie Dalzell. Send your subscriptions to:

CoCo-Link
31 Nedlands Crescent
Pt. Noarlunga Sth.
South Australia
Phone: (08) 3861647

The following article was submitted by a User Group member who has experience with the MS-DOS operating system. He wrote the following article to supply information to a PC users group. You will notice that it was written from the viewpoint of an MSDOS user. Editorial changes and additions appear surrounded by square brackets [...].

All criticisms should be redirected to /Nil.

COCO Windows VS Windows 3.0

by George McLintock

As a hobby type programmer I often take an interest in current fads and recently got a copy of [MS-DOS (c)] Windows 3 for a look at. I was rather disappointed with the results

The package itself (for around $175 locally) provides no access to any of its features that you can use in your own creations. [Yes, you can ... see the next sentence]. To do a simple thing like positioning a system window on the screen from within your own program, you will need to spend another 700 odd dollars to get a software developers kit as well. This is not hobby type money so I can't comment on what you might get if you buy that as well.

I have a much older multi tasking/multi windows type operating system that runs on my old computer (OS9 on a Tandy Color computer, which predates OS/2 by at least 10 years) and some comparasion between it and the 'new generation' windows might be of interest.

A simple direct comparasion of the CPU/software overheads associated with multi tasking on the two systems is as follows. The reduction from the native mode value is a measure of these overheads.

	Windows(386)	OS9(CoCo)
Native Mode (DOS on 386)	100	18.77
Multi tasking/windows		
Single task	33	18.77
Two tasks	29	18.69
Three tasks	30	18.58
Four tasks	32	18.40
Five tasks	33	18.27
Six tasks	33	18.10

These relative weights were obtained from a simple compiled Basic program on each machine that counts the number of times a common loop is executed per minute of actual elapsed time. (CPU fully occupied). The same program was then run as multiple tasks to produce a separate count for each task which were then added to obtain a total count for the specified number of tasks. The reduction in total count for multiple tasks is a measure of the cpu/software overheads of multi tasking. I am aware of the problems with this sort of comparasion, and will ignore all criticisms of the metholodogy

Windows 3 incurs an enormous penalty for the first task in a multi tasking environment, but after that it gives some erratic results. (5 tasks give the same count as 1, which is significantly higher than 2 or 3 tasks). I have no idea why. OS9 does its multi tasking with standard time slicing and the results are as you might expect. Native mode with OS9 is a single task, single window environment anyway. The original CoCo mode has interpreted Basic only.

Some knowledge of the two systems involved will make the comparasion even more interesting. The windows values were obtained from a standard type 386 SX (16 MHz clock, landmark around 21 MHz) with 8 Megs of memory. Multiple tasks were run as DOS programs in their own window, started from the keyboard, with the total count per minute derived from whole minutes with no keyboard or mouse activity. Rather painful but can be done. I apologise to those unfamiliar with the general background of the 386 system.

The original Tandy Color Computer (CoCo) is older than the original IBM PC (by at least a year). The model used here (CoCo 3) was released in 1986. It uses a 1.66 MHz 6809 CPU. (and yes there is a decimal point between the 1 and the 6. The original CoCo is 0.83 MHz). The 6809 is from Motorola, has an 8 bit data path, 16 bit internal registers [Intel's 8088 CPU has a similar system] and 16 address lines. It uses a separate memory management unit to address up to 512K [now 1 Mb] in 8K blocks of actual memory, with a maximum of 64K mapped in at any one time. (1 meg upgrade kits from third party suppliers). For comparasion, the same program under DOS on a 4.7 MHz 8088 produces a relative count of 13.2.

I would have liked to try some other comparasions as well, but my other CoCo equipment is a bit old to do anything meaningful. eg the disk drives were made in 1981, and are single sided with a 30 ms track to track stepping rate. Rather slow, but I got them second hand and they satisfy my requirements. They also demonstrate that Tandy is just as good at maintaining backwards compatability as IBM ever was. Some form of comparasion with multi tasking graphics is probably feasible (the CoCo has EGA type graphics), but without programming access to the windows operating system I haven't been able to think of one.

OS9 is a real-time [multi-user] multitasking operating system written for the Motorola 68XX[X] family of processors, and more lately available for 80386 Intel processors as well. Multitasking is standard, all compilers produce re-entrant code, modules are linked at run time, all I/O is through standard interfaces by means of device drivers, all graphics modes are supported etc. It provides very efficient use of memory and 512K is a reasonably large system. Only a single copy of a module is required in memory for the use of multiple processes. Coding by OS9 programmers is in general much tighter than that of other systems. For example, the Basic09 runtime interpreter is only 12K. Compared with 78K? for GWBasic. Each source module can be compiled separately. If it calls another module at run time the system will automatically link to that other module as required. If the other module is not in memory, it is automatically loaded so that the link can be established. A single module can be used by any number of other modules at the same time. The link is established simply by calling the routine.

As another trivial and simplistic comparasion, the Quick Basic runB module is approx 77.5K and the count program compiles to approx 6.7K. Apart from its other overheads, Windows 3 loads a total of 84K (77.5 + 6.7) of program code for each task. The OS9 equivalents are 12K for the runB module and 410 bytes (no K's) for the count program. OS9 loads 12K + 410 bytes for the first task and then adds an extra 64 bytes in the system page for each additional time the task is run.

OS9 Level 1 (with a 64K memory limit) was released for the CoCo in 1983. Level 2 (any amount of memory) was released with the CoCo 3 in 1986. Around 1987 Tandy released a standard GUI (graphics user interface) called Multiview for OS9 which appears very similar to Windows. For a non-technical user they would appear to be much the same. Software packages come with the equivalent of a Window's .PIF file which will automatically set them up to run under Multiview. You drive the system with a mouse, joystick (emulating a mouse) or keyboard. Icons etc follow the same general style as windows.

OS9 windows are proper system type windows with the basic unit being device windows that can be text or graphics. You can have any number of these [within limits], of any size, located anywhere on the physical screen, as well as a number of different physical screens accessible by a screen change 'hotkey'. They are standard OS9 devices and a program can direct its output to any device open to it. Output can be automatically scaled to match the actual window size. Device windows cannot overlap each other on the physical screen and all must be the same type (text or graphics) at any particular time. Graphic device windows can be any resolution and can be mixed, within the limits of the physical display hardware. eg a program using a 4 color screen can have a device window on the same physical screen as a program using a 16 color device window, within the limits of no physical overlap etc. The physical screen is of course set to 16 color mode in these circumstances, with some practical limitations on sequences etc.

Overlay windows sit on top of device windows and can be located anywhere within it. The normal procedure is to start a process in a device window which belongs to it. Different modules within that process can then use overlay windows to locate their output anywhere within it. However, programs can send their output to different device windows and run overlay windows in them. Assuming the owner of that other device window (if it has one) allows it. A process owns a device window if it is running under a separate shell within it, otherwise it is open to all comers. The system provides a standard set of clipboard type buffers to allow for the transfer of data between processes. These can be used without any direct linking between processors at run time.

These general concepts don't seem to apply too well to Windows 3. In one way it seems to operate as a single full screen device window with each task having its own full screen overlay. A better alternative might be to consider it as allowing device windows to overlap each other, with each task having its own full screen device window overlaying the one below it. This might at least make some sense for the requirement to re-install the whole system to change the graphic display

mode. I expect it would be logically very difficult to overlay graphic device windows with different resolutions.

Windows 3 does allow you to change the size and location of the window for a task by using the mouse to drag the edges. However, when you do this you are not actually changing the window itself, position 0,0 remains in the top left corner of the physical screen so it is not even a true overlay window either. All you appear to be doing is to change a Basic type view of the actual window which continues to occupy the full physical screen.

Apart from its age and CPU performance, the big difference between these systems is price. The CoCo was discontinued outside the USA a couple of years ago, but is still available in that country. Some prices ($US) from a Nov 1990 magazine are : OS9 Operating system, including Basic compiler and good manuals - $72. Multiview, Windows 3 type standard GUI - $45. OS9 Development system, includes Assembler, programming tool box etc - $90. Common packages like word processors, data bases, spreadsheets etc have a similar sort of price relativies to their DOS equivalent. A two floppy, half meg memory system with color monitor (Amiga analogue style) - $520. Hard drives break the pattern, a 40 Meg hard drive - $580.

The CoCo under OS9 provides very good performance for its age and price, but it is largely unknown outside its own circle of users. OS9 does have a big brother, called OS9/68000 which runs on Motorola 68000 series CPU's. I have no idea where it fits in any hierarchy of Unix systems for that chip. There are versions available for the Atari ST and Amiga low end machines. Its performance is highly regarded by ex OS9 users on CoCos.

A couple of interesting new machines were released in the USA late last year (1990) which have taken different approaches to an upgrade path for existing CoCo users. The installed CoCo base is around 3-4 million, so even if you haven't heard of it, and Tandy have ignored it, it is not completely insignificant for smaller companies in the computer world. Both these machines are based on the 15 MHz 68070 chip (Signetics) using the OS9/68000 operating system. I haven't seen any performance figures for either machine, but they should be impressive.

The cheaper of these machines makes a complete break with the CoCo but offers source level compatability with OS9 software and similar operating systems. For $860 (US) you get a base system that includes a 15 MHz 68070, 1 Meg memory, a single 1.4 Meg floppy, 720 x 540 x 256 graphics (with separate graphics co processor), OS9/68000 operating system in EPROM (or boot from disk), Basic and C compilers plus a set of standard software packages and utilities, including read/write MS-DOS disks. You have to add a standard type analogue monitor (Amiga type frequency). You also need another board ($350) to expand with more memory and hard drive (SCSI) etc. But still very competitively priced for the performance. The basic unit includes a network controller to attach up to 120 other units at 100K bauds and is obviously aimed at the school small business market as well. The CoCo still has a reasonable presence in these areas in the USA.

The second, and more expensive base machine provides full backwards compatability with the existing CoCo line (hardware and software). It does this with a 2 CPU system. With a 6809 only it operates as a CoCo and uses all existing hardware and software. (Basic system price around $500 US). As a two CPU system it adds a 68070 board running OS9/68000 with the 6809 used as a separate I/O/graphics co processor. Basic system price is around $1400 US including 1.5 Meg memory and other features, software etc as per the other 68070 system. This one is upgradeable without any extra boards, to 16 Meg memory, SCSI hard drive and you retain the option of still using it as a stand alone 6809 system. The two CPU upgrade path has been tried before without success, but who knows. Anyone remember the Tandy Model 16, a Z80 for CPM and a 68000 for Unix. This new one at least has much the same operating system on both chips.

Needless to say the ads for both these systems are not too specific on details and I haven't yet seen any reviews of them. It is a fact of life that you are unlikely to see any mention of them outside the specialist OS9/CoCo magazines either. According to rumour, at least one of these machines will be available in Australia later this year from one of the remaining CoCo third party suppliers here, so you may hear about it again. There are still a few CoCo/OS9 fans around who would be interested.

In any case it is good to know that there are still some companies around who can offer reasonably priced high performance software and hardware. The pricing structure (and other aspects) of Windows 3 (and OS/2) reflect all the worst features of a monopoly (cartel) supplier, and I would like to see something like OS9/68000

be at least successful enough on a reasonably priced machine to maybe do for operating systems what Borland's Turbo series did for compilers. Unix is frequently mentioned as a competitor for Windows/OS2, but the major potential suppliers of Unix for Intel machines appear to be no better with either pricing or hardware requirements. I certainly wish these other suppliers every success and assuming realistic Australian pricing (ie no Microsoft style inflation factor in the exchange rate conversions) will probably do my bit to support one of them.

oooooooooo0000000000oooooooooo

Basic09 Parameters
By Bob Devries

Did you know that there are two ways to pass parameters to a basic09 procedure? Yep, here they are:-

1. By **REFERENCE**
2. By **VALUE**

The first is the normal, default method, but the second way can be used easily, too. Consider the following example, a little procedure to change a string variable to uppercase.

```
(* toupper - change case of string 'line' *)
PARAM line:STRING
DIM x,y:INTEGER
DIM upline:STRING
DIM char:string[1]
upline=""
FOR x=1 TO LEN(line)
  char=MID$(line,x,1)
  IF char>=" " AND char<="Z" OR char>"z" THEN
    upline=upline+char
  ELSE
    upline=upline+CHR$(ASC(char)-32)
  ENDIF
NEXT x
line=upline
END
```

Now, if you wanted to use this procedure as part of your programme, you would normally do this:-

```
fox="the quick brown fox"
RUN toupper(fox)
PRINT fox
```

You would see that the variable 'fox' has been changed to uppercase characters. If, however, you called the procedure like this:

```
fox="the quick brown fox"
RUN toupper(fox+"")
PRINT fox
```

You would see that there was no change to the variable 'fox'. This is because adding the **+""** to the line, makes it an expression, and this forces Basic09 to calculate the value (even with string variables) and pass the value to the procedure.

You might say, 'what is the use of doing that?'. Well, I'll tell you. There might be a programme in which you wish to be able to change the contents of a passed variable without affecting the calling procedure's variable, without having to set up a temporary variable in the first procedure, which would take up extra memory.

The same rules apply to all the other variable types, too. For instance, here's some more examples:-

```
RUN hammer(nail+5) (* here nail is INTEGER *)

RUN ask(price+1.5) (* here price is REAL   *)

RUN hex(LAND(num,$0F)) (* here num is BYTE *)
```

So you can see, that there some useful ways to use Basic09 parameters besides the 'normal' way. Think about it.

Regards,
Bob Devries.

oooooooooo0000000000oooooooooo

File De-fragmentator
by Bob van der Poel
comments by Bob Devries
Part Two

Here is part two of Bob van der Poel's programme for rearranging your disk files into contiguous sectors. Our thanks go to Bob for supplying this free of charge and in the Public Domain. As I mentioned last month, you will need the Kreider C Library to compile this gem, and, as usual both that and this programme are available from our PD Library for the usual copying fee. Please note that requests for PD software now go to Jean-Pierre Jacquet at the address on the front of the newsletter.

Make sure you append this part to the end of part one before you try to compile it.

Regards,
Bob Devries.

```
/*============================================================
   Main workhorse. This routine reads the directories and checks
   the status of all the files. Any fragmented files will be
   copied and renamed. If any directories and the "do subdirs flag"
   is set this routine will be recursively called.
*/

process(dir)
char *dir;
{
   char *dirnames,*next;    /* ptrs to malloc filenames  */
   int numnames,nsegs;      /* working vars              */
   register int t;
   int ipath;               /* os9 file pathnumbers      */
   char tempstr[20];
   struct fildes fdbuf;     /* struct for FD sector      */

   if((chdir(dir))!=NULL) quit("Unable to access directory");

   dirnames=next=getdir(&numnames);  /* read directory */

   for(;numnames;next+=30,numnames--){   /* process each filename */
      fputs(next,stdout);
      space(40-strlen(next));

      if((ipath=open(next,1))>0){
         totalfls++;
         _gs_gfd(ipath,&fdbuf,256);  /* read RBF file descriptor */
         nsegs=countsegs(&fdbuf);
         if(nsegs<2){
            fputs("Okay\n",stdout);
            close(ipath);
         }
         else{
            totalfg++;

            if(report){ /* don't fix things, just report */
            fputs("Fragmented: ",stdout);
            itoa(nsegs,tempstr);
            puts(tempstr);
            close(ipath);
            }
            else{  /* copy/delete/rename file */
               fputs("Processing: ",stdout);
               copy(ipath,&fdbuf,next);
            }
         }
```

```
      }
      else(   /* if unable to open file then try to access
                 as a directory  and process the next level. */

         if((chdir(next))==NULL){
            fputs("Directory...",stdout);
            if(dodirs){
               puts("processing next level");
               process(".");  /* do the next level down */
            }
            else puts("not processed");
            chdir("..");  /* backup to the parent */
            if(dodirs){
               space(40);
               puts("subdirectory complete");
            }
         )

         else(   /* not a directory...skip it then */
            puts("Can't open, skipped");
         }
      }
   }
   free(dirnames);  /* return memory for directory list */
}

/* ==========================================
   Copy a fragmented file in the hope that pre-extension will fix things..
*/

copy(ipath,oldfd,filename)
int ipath;              /* already open original file path */
struct fildes *oldfd;   /* file descriptor structure       */
char *filename;         /* name of file being copied       */
{
   int opath,nsegs;
   register int t;
   struct fildes newfd;
   char tempstr[100];

   /* create new file, abort if we can't  */

   if((opath=ocreat(tempfile,_WRITE,oldfd->fd_att))==-1)
        quit("Unable to create temporary file");

   /* pre-extend file size, abort if no room */

   if((_ss_size(opath,oldfd->fd_fsize))==-1)
        quit("Not enough room for copy on disk");

   while((t=read(ipath,copybuf,BUFSIZE))>0){
      if(write(opath,copybuf,t)!=t)
         quit("Write error during copy");
   }
   if(t)  /* copy complete, if t!=0 there is a read error */
      quit("Read error during copy");
```

```
   /*  check new fragmemation.. */

   _gs_gfd(opath,&newfd,256);  /* read RBF file descriptor */
   nsegs=countsegs(&newfd);
   if(nsegs>1) puts("Still fragmented!");
   else{
      puts("okay");
      totalfix++;
   }
   close(ipath);

   /* delete oldfile */
   if(unlink(filename)){
     cont("Unable to delete original file");
     goto copyx;
   }
   /* rename the newfile */
   strcpy(tempstr,"rename ");
   strcat(tempstr,tempfile);
   strcat(tempstr," ");
   strcat(tempstr,filename);
   if(system(tempstr)){
     cont("Unable to rename new file");
     goto copyx;
   }
   _ss_pfd(opath,oldfd);            /* set dates to match old */
copyx:
   close(opath);
}

/* ======================================
   Display message and ask if okay to continue. Terminate if answer is not <Y>.
*/

cont(msg)
char *msg;
{
   register int c;

   fputs("\n\n      ",stdout);
   puts(msg);
   fputs("       Okay to continue (y/n): ",stdout);
   c=getchar();
   if(toupper(c)!='Y') quit("");
   puts("");
}

/* ==========================================
Create a list of all the entries in the current directory.
This uses memory allocated via malloc(). The pointer returned points to the first entry; count==number of
valid entries (not including . and ..).
*/

getdir(count)
int *count;
{
   DIR *dirp;
```

```
   register char *nbuf,*next;
   long _gs_size(),dirsize;
   if((dirp=opendir("."))==NULL) quit("Unable to open directory");
   dirsize=((_gs_size(dirp->dd_fd)/32)-2)*30;
   if((nbuf=malloc((int)dirsize))==NULL) quit("Out of memory");
   for(*count=0,next=nbuf;entry=readdir(dirp);){
      if(isalpha(entry->d_name[0])){
         strcpy(next,entry->d_name);
         *count+=1;
         next+=30;
      }
   }
   closedir(dirp);
   return nbuf;
}

/* ============================================
   Universial terminate routine. Prints error message and quits.
*/

quit(s)
*s;
{
   fputs("\n",stderr);
   fputs(s,stderr);
   fputs("\n",stderr);
   fputs("UNFRAG terminated\n",stderr);
   exit(errno);
}

/* ================================================
   Output 1 to n spaces
*/

space(n)
int n;
{
   do{
      fputs(" ",stdout);
   }while(--n>0);
}

/* ==================================
   count  the number of segments in the current file the file descriptor buffer must have been  read  and
stored at fdbuf.
*/

countsegs(fdbuf)
struct fildes *fdbuf;
{
   register int t;
   int nsegs;
   for(nsegs=t=0;fdbuf->fdseg[t].size;nsegs++,t++);
   return nsegs;
}
```

oooooooooo0000000000ooooooooooo