AUSTRALIAN OS9 NEWSLETTER

EDITOR :
Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

(07) 345-5141

AUGUST 1989

ATTR Syntax: Attr filename [permissions] Usage : Examine or change the security permissions of a file Opts: -perm = turn off specified permission perm= turn on specified permission -a = inhibit rms : d - directory file s - nc to owner w - write permit to own r - read permit to public pw - te permit to public BACKUP Syntax ge : Copies all data from one de ead error occurs s = single rites BASIC09 Syntax : Basic0 ge BUILD Syntax: Build filenar s from standard input CHD S nge working directory to specifi > Usage: Change execution directory to specified path CMP Syntax: Cmp filename1 filename2 Usage : File comparison utility COBBLER Syntax: Cobbler devname Usage : Creates OS-9 bootstrap file from current boot CONFIG Syntax n disks COPY Syntax data from one fil E Syntax : Date [t Opts: t = specify ame> Usage : Check directory for wor sters -m = save of unused cluster nly -o = print <devname> }<devn : Del [-x] filenam s : -x = delete x: Deldir directo yntax: Dir [e] [x] the file names y x=print executi] Usage : Display s converted characters to standard output DSAVE Syntax : Dsave [-opts] [dev] [pathname] Usage : Generates procedure file to copy all files in a directory system Opts : -b make a system disk by using OS9boot if present -b=<path> = make system disk using path = indent for directory levels = do not process b makdir command o num K ECHO Syn tandard output ED oriented text edito s text error messages for given error numbers EX Syntax: ex <modname> Usage: Chain to the given module FORMAT Syntax : Format <devname> Usage : Initializes an OS-9 diskette Opts ; R - Ready L - Logical format only "disk name" 1/2 number of sides 'No of

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group

EDITOR : Gordon Bentzen

HELPERS : Bob Devries and Don Berrie

SUPPORT : Brisbane OS9 Level 2 User Group.

August, 1989

Let's talk about utility programmes. Hands up those of you who quickly write a programme in (dare I say it?) Disk Extended Colour Basic, to do a particular job for a particular time, and then promptly turn off the computer when that job is done ? Quite a few of you I'll bet. I know I do it all the time. Have you ever considered that you have just written, and destroyed, a programme that might be useful to someone else ? And above all, with a little bit of extra thought, could be written in Basic09?

Oh sure, I can hear you say, why would I turn on the computer, boot up OS9, and then load up Basic09 just to run a ten line programme ? Well, I am here to say that it is a very good idea to try it anyhow. There is a programme elsewhere in this issue, written by Bob Devries, that started out life just as a little half-a-dozen lines to find the file descriptor information of a particular file. The information required was the creation date of a source-code listing, and, as we all know, the DIR E command only tells us the date of last modification, not its creation. So, having found out how to read the file descriptor, Bob put this to further work and after a few hours work, out popped 'UpperDir' a Utility Programme to rename all directory files to uppercase and all normal files to lower case.

You know, I'd like to hear from other people who have perhaps put together a few ideas like that to do a task and, even if it does not work as expected, send it to us here at the newsletter, and we may be able to add some more code to make it work better, or do more yet. Sometimes, I find that I start writing a programme to do something, and find when I am part-way through that I can't finish it because of lack of experience or not enough knowledge of the subject. What happens then is that usually it gets forgotten, or picked up again much later. Sometimes I wonder if someone else could perhaps continue with it to make it turn out correctly. Of course, the best idea would be to talk with people at the local user group, but that is usually still three weeks away, and I need help NOW! When this happens to you, drop us a few words on paper (or disk) and we will endeavour to set you straight if possible. Don't forget, we three are not super programmers either, so don't expect miracles, and please give as full an explanation of what you want as you can.

Well, I guess it's time to remind you about your subscriptions. They are due at the end of THIS month. Please get those cheques away as soon as possible so that we can decide whether it was all worth it. I for one have learnt a lot from writing in this newsletter, and from having to read the manual whenever I had to cover a subject I was not too sure about. I know I am speaking for the three of us when I say that I'd like to continue for another year, so how about it fellows ? Make my day, send that cheque!

Topics covered in this month's newsletter include 'UpperDir' by Bob Devries, more explanations about the 'Edit' utility (that word again) by Don Berrie, and also of course, more of Bob's Database in 'C'. I'd like to see more questions coming to us from you people out there, even if you've found out the answer yourself in the mean time, don't forget that others may have similar problems. At a local user group recently, I showed some-one how to use the Config programme to set up a double sided OS9 system disk (Oh, don't blush, friend). I'm sure there are others who have the same problems, including difficulties in setting up Multi-Vue and other programmes.

Well, that's about all for this month, happy computing,

Regards,
The Editor.

MACROS: Continuing a Tutorial for the OS9 Macro Text Editor

Two of the most overlooked features of the Macro Text Editor are its ability to handle more than one file at a time, that is multiple buffers, as well as its ability to handle quite complex macros. It is not really surprising that these features are overlooked, because the manuals (both Level 1 and Level 2) leave much to be desired in their explanations. First of all, let's look at multiple buffers.

Why do we need multiple buffers? Well consider the following. We are writing an article for the newsletter using good old EDIT, and we decide that, as an example, we want to include our startup file into the text to illustrate a point. Ever tried it? If, like most people that I know, you looked in the manual to see how to do it, couldn't understand it, and gave up and used Stylograph; read on.

The other (and I guess the most important) reason for having the ability to edit multiple files is of course for our friends with Level 1 systems, or for those with Level 2 systems which don't support windowing. (Yes, some Level 2 systems don't have that ability.) Back to our example. For the purpose of this tutorial, we decide that we want to edit the text before we import it into our document. To do this, we load the new (ie /dd/startup) file into a DIFFERENT buffer. (We could have imported the whole file into our document, and then edited it, but that doesn't serve the purpose of this example.) We do this by switching to a new buffer :

B0

When EDIT is started, by default there are two buffers already defined, the primary buffer (denoted by *), and the secondary buffer (denoted by $). At startup, the primary buffer is always buffer 1, and the secondary buffer is buffer 0. So the command B0 makes buffer 0 into the primary buffer. You can only operate on the text contained in the primary buffer. Rather than make buffer 0 the primary buffer, you could have equally typed :

B6

and created buffer 6 (if it did not already exist), and made it the primary buffer. I don't know how many buffers you can create, but you are limited (initially at least) only by the amount of memory available. Remember the [#nK] command line modifier from last months installment. To display the buffers currently defined, type :

.DIR

(Don't forget the full-stop.) This will show you a list of the currently defined buffers, with the primary buffer marked with an asterisk, and the secondary buffer marked with a dollar sign. It will also show a list of MACROS currently defined. We will come to those later.

Next, we will need to load the text into the new primary buffer. This is a two step process. The first thing to do is to tell the editor the new filename. This is where the confusion starts. To do this you need to use the :

.READ "pathlist"

command. The parentheses are necessary. Therefore our command line becomes (in my case at least) .READ "/H0/startup". So far we have only told the editor the name of the file that we wish to edit. Next we must actually read the file into the buffer. To achieve this we use the :

Rn

command, which causes the editor to read n lines. (eg R* reads the whole file into the buffer, if it will fit). As the editor is loading text into the buffer, is displays the lines as they are read. From here on, we edit the file as per normal.

When we have the text in the format we desire for incorporation into the original document, we use the :

Pn

command. This is where the concept of defined primary and secondary buffers becomes important. The Pn command PUTS n lines of text into the secondary (in our case buffer Ø, where we just were) buffer. Remember, that the buffer last used, becomes the secondary buffer. (Use the .DIR command if you are not sure where you are.) If we were to switch to the original document using the B1 command, we could then load text from the secondary buffer (now buffer Ø, the last buffer that we used), by the use of the command :

Gn

This command GETS n lines from the secondary buffer. The thing to remember, is that both G and P operate only between the primary and secondary buffers, whatever their numbers may be.

Other commands that you may wish to use are the W command, which writes n lines from the current buffer to the pathname as defined in the initial command line, or by a .READ command. The Vn command turns on (n=Ø) or off (n=any other number) the verify command. This will prevent the editor from echoing lines to the terminal. You need to exercise care with the use of the V command, because it will also prevent listing of your document to the screen using the Ln command!

As with the previous article in this series, you may use both upper of lowercase for these commands.

Next month, we will discuss the editing of files larger than the workspace, and start on the concept of MACROS, and hopefully we will provide a macro to delete lines of text and display the next line in the file, rather than the one just deleted (a quirk of this editor which I find absolutely infuriating!!). Until then keep up the good work.

oooooooooo0000000000ooooooooo

A Database in C

By Bob Devries

Here are two more parts to my C Database programme. The first is 'help.c' and provides a help screen documenting all the commands available, and the second is the 'find.c' routine which allows you to find a particular record in the file.

The 'help' command clears the screen with a call to the 'cls' function, and prints a series of lines to the screen, and then waits for a keypress. When a key is pressed, it clears the screen and returns to the main programme.

The 'find' function prints up an empty shell for the data entry, and then asks for the surname to be found. It then goes looking for the record that matches that name. If found, it is displayed, and that record becomes the current record. If it is not found, 'Not Found' is printed on the status line, and the last record is displayed. Note that the programme is very litteral in its search, and does differentiate between upper and lower case. So make sure you get it right. Perhaps someone can devise an easy way to overcome this limitation.

```
help()
/* pressing h from the data screen will clear the screen and display the */
/* help message, then wait for a keypress. the main() function then re- */
/* displayes the data screen. */
{
        char ch;
        ch = '\0';

        cls();
        printf("\n\n\n     a = amend (edit) displayed record.\n");
        printf("     p = goto previous record.\n");
        printf("     n = goto next record.\n");
        printf("     i = insert new record.\n");
        printf("     d = delete displayed record.\n");
```

```
        printf("     f = find a record by surname.\n");
        printf("     m = find another record with same criteria.\n");
        printf("     h = help (this screen).\n");
        printf("     e = exit from programme.\n");
        printf("\n\n\n     Press any key to return...");
        ch = getch();
        cls();
}

find()
/* find() locates the first occurence of the surname which is input and */
/* returns the record number of that record */
{
        char nametmp[21];       /* temp surname variable */
        int temprec;            /* temporary current record */
        long lof;

        temprec = 0;
        fseek(fp,01,2);
        lof = ftell(fp);
        cls();
        scrnmask();             /* display empty screen mask */
        cursor(10,23);
        eraselin();             /* erase message line */
        cursor(10,23);
        printf("Input surname to be found.");   /* print prompt */
        cursor(14,5);
        gets(nametmp);          /* get surname to be found */
        cursor(10,23);
        eraselin();             /* erase message line */
        do {
                temprec += 1;
                fseek(fp,(long)(temprec - 1)*sizeof(mail),0);
                if (ftell(fp) < lof)    /* read until eof */
                        {
                        fread(&mail,sizeof(mail),1,fp);
                        }
                else
                        {
                        cursor(10,23);
                        eraselin();
                        cursor(10,23);
                        printf("Not found.");
                        break;
                        }
        } while ((strcmp(nametmp,mail.surname) != 0));
        return(temprec);        /* return record number of wanted file */
}                               /* or last record if not found */
```

oooooooooo0000000000oooooooooo

LABEL PRINTER This is another Basic09 programme submitted by Phil Frost of Kalgoorlie. Phil states "the label_Printer programme arose from the need to re-label disks after a re-organization of my collection." We thank you Phil for sharing this programme with us, and no doubt other members will also find it useful. The programme will help you print labels to attach to disks, and contains all the necessary instructions when run. I note that Phil has included printer codes for a Tandy DMP-130, however his commented source will make it very easy to alter the codes to suit any other printer.

```
PROCEDURE label_printer
(* One Line Printer programme for 3 1/2 x 15/16 labels
(* For Printing Lables
(*
(* by Phil Frost
(*
(* The Printer Codes are Set for a DMP-130 Printer
DIM printer:BYTE
DIM a$:STRING[35]
DIM b$:STRING[42]
DIM c$:STRING[60]
DIM select:BYTE
(*
OPEN #printer,"/p":WRITE
(*
50 PRINT CHR$(12);
PRINT  \ PRINT  \ PRINT
PRINT TAB(33); "LABEL PRINTER."
PRINT  \ PRINT
PRINT TAB(23); " 1/ Normal     35 Characters per Line"
PRINT
PRINT TAB(23); " 2/ Compressed 42 Characters per Line"
PRINT
PRINT TAB(23); " 3/ Condensed  60 Characters per Line"
PRINT
PRINT TAB(23); " 4/ End "
PRINT  \ PRINT
PRINT TAB(33);
INPUT "Select",select
ON select GOTO 200,300,400,500
200 PRINT
PRINT TAB(10); " Type quit or QUIT to end input"
PRINT TAB(10); " 35 Characters Per Line"
PRINT TAB(10); " ***********************************"
PRINT TAB(9);
INPUT a$
IF a$="quit" OR a$="QUIT" THEN
GOTO 50
ELSE
ENDIF
PRINT #printer,CHR$(27); CHR$(19);
PRINT #printer,a$
GOTO 200
300
PRINT TAB(10); " Type quit or QUIT to end input"
PRINT TAB(10); " 42 Characters Per Line"
PRINT TAB(10); " ******************************************"
PRINT TAB(9);
INPUT b$
IF b$="quit" OR b$="QUIT" THEN
GOTO 50
ELSE
ENDIF
PRINT #printer,CHR$(27); CHR$(23);
(* set compressed character
PRINT #printer,b$
```

```
GOTO 300
400
PRINT TAB(10); " Type quit or QUIT to end input"
PRINT TAB(10); " 60 Characters Per Line"
PRINT TAB(10); " ************************************************************"
PRINT TAB(9);
INPUT c$
IF c$="quit" OR c$="QUIT" THEN
GOTO 50
ELSE
ENDIF
PRINT #printer,CHR$(27); CHR$(20);
(* set condensed character
PRINT #printer,c$
GOTO 400
500 PRINT #printer,CHR$(27); CHR$(19);
(* reset printer to normal mode
CLOSE #printer
END
```

oooooooooo0000000000oooooooooo

The Public Domain Library.

We have received a number of requests from members for a complete listing of the Public Domain files of the National OS9 User Group. The task of providing this however, is not as simple as it sounds. Bob Devries reviewed some 34 archived files in last month's newsletter which provided information about each of the files. This month we include a simple directory listing of another disk from the library which includes many files published in past newsletters. Please refer to this newsletter listing if you want any of these files, and order in the usual way. i.e. Post formatted disks with return postage plus $2.00 copy charge for each disk, and of course, specify the files or other disks required.

```
Directory of /d2/ASM/CMDS
  util       dmode      mverify    mrename    ar09       ar
-------------------------------------------------------------------------------
Directory of /d2/ASM/HELP
  ar.doc
-------------------------------------------------------------------------------
Directory of /d2/C_MW/CMDS
  lsn0       database    lb          ls          rdump       hxd
-------------------------------------------------------------------------------
Directory of /d2/C_MW/SRC
  lsn0.c       database.c    ansi.h        lb.c          rdump.c
  hxd.c
-------------------------------------------------------------------------------
Directory of /d2/C_MW/HELP
  most_c.2      os9hxd.art
-------------------------------------------------------------------------------
Directory of /d2/BASIC09/CMDS
  zap             ohms            label_printer    upperdir
-------------------------------------------------------------------------------
Directory of /d2/BASIC09/SRC
  zap             iconedit        ohms_law         label_printer
  upperdir.B09
```

```
Directory of /d2/SHELLSCRIPT
  cc1_cc2      cc1_patch
-------------------------------------------------------------------------------
Directory of /d2/SHELL
  shellplus            shellplus.doc        shellplus.install
  loglock.on.scr       loglock.off.scr      wild.on.scr
  wild.off.scr         noblock.on.scr       noblock.off.scr
*>SHELLPLUS.2.1
-------------------------------------------------------------------------------
Directory of /d2/SHELL/SHELLPLUS.2.1
  shellplus            shellplus.doc        shellplus.install
  atcheck.on.scr       atcheck.off.scr      loglock.on.scr
  loglock.off.scr      wild.on.scr          wild.off.scr
  noblock.on.scr       noblock.off.scr      shellscript
  param.a              param.bin            sdate.a
  sdate.bin            prompt.bin
```

oooooooooo0000000000oooooooooo

UpperDir, A Basic09 Utility.
By Bob Devries.

I've had heard several times in the past the complaint that it is impossible to tell files and directories apart in OS9. The answer of course lies in good housekeeping, and by having all files in lowercase characters, and all directories in uppercase characters. Well, that's all very good, but what if you've already got them all in uppercase like the OS9 level one disks as they were supplied by Tandy? Of course a utility programme is the answer. And here it is. I have called it UpperDir.

It opens the current working directory, and reads the filenames one by one, and decides whether they are files or directories, and renames them using lowercase or uppercase as is necessary. If they are already correct then no changes are made.

Please note that this programme requires that you have 'Rename' in memory or the current execution directory, or it will not work. If you forget this, the UpperDir will terminate with an error.

I have also written a version in 'C', which does not require 'Rename' to work, but does the renaming itself. I will be presenting that version in a future newsletter.

```
PROCEDURE UpperDir
 0000      (* UpperDir - CopyRight (c) 1989 By Bob Devries
 002F      (* Freely distributable.
 0047      (* UpperDir reads the current directory, and changes all
 007F      (* the directory names to UPPERCASE, and all the files to LOWERCASE
 00C2      (* It uses the 'Rename' utility, so be sure it is in memory
 00FD      (* or in the execution directory.
 011E      (* Version 1.00 Date 12th July, 1989.
 0143      BASE 0
 0145      TYPE direct=fname:STRING[29]; fsect(3):BYTE
 0160      DIM entry:direct
 0169      DIM dirpath:BYTE
 0170      DIM path:BYTE
 0177      DIM name,newname:STRING[29]
 0187      DIM attr:BYTE
 018E      DIM changed:BOOLEAN
 0195
 0196      attr=0
 019D      name=""
 01A4      newname=""
 01AB      OPEN #dirpath,".":READ+DIR
 01B7      GET #dirpath,entry
```

```
01C1      GET #dirpath,entry
01CB      WHILE NOT(EOF(#dirpath)) DO
01D6 1      GET #dirpath,entry
01E3        IF LEFT$(entry.fname,1)=CHR$(0) THEN
01F6          GOTO 30
01FA        ENDIF
01FC        RUN extname(entry,name)
020B        ON ERROR GOTO 10
0211        OPEN #path,name:READ
021D        ON ERROR
0220        GOTO 20
0224 10     ON ERROR
022A        ON ERROR GOTO 15
0230        OPEN #path,name:READ+DIR
023C        ON ERROR
023F        GOTO 20
0243 15     PRINT "File "; name; " inaccessable."
0264        ON ERROR
0267        name=""
026E        newname=""
0275        GOTO 1
0279 20     RUN readfd(path,attr)
028B      EXITIF attr=-1 THEN
0298        PRINT "Error in Syscall 'GetStt SS.FD call'"
02C0      ENDEXIT
02C4        changed=FALSE
02CA        IF LAND(attr,$80)=$80 THEN
02DB          RUN upper(name,newname,changed)
02EF        ELSE
02F3          RUN lower(name,newname,changed)
0307        ENDIF
0309        PRINT newname,
030F        IF changed=TRUE THEN
031A          SHELL "rename "+name+" "+newname
0331        ENDIF
0333        CLOSE #path
0339        name=""
0340        newname=""
0347 30   ENDWHILE
034E      CLOSE #dirpath
0354      PRINT  \ PRINT "Finished !"
0364      END
PROCEDURE extname
0000      TYPE direct=fname:STRING[29]; fsect(3):BYTE
001B      PARAM entry:direct
0024      PARAM name:STRING[29]
0030      DIM i:INTEGER
0037
0038      FOR i=1 TO 28
0048      EXITIF ASC(MID$(entry.fname,i,1))>127 THEN
005E        name=name+CHR$(ASC(MID$(entry.fname,i,1))-128)
0078      ENDEXIT
007C        name=name+MID$(entry.fname,i,1)
0091      NEXT i
009C      END
PROCEDURE upper
0000      PARAM name:STRING[29]
```

```
000C      PARAM newname:STRING[29]
0018      PARAM changed:BOOLEAN
001F      DIM i:INTEGER
0026      DIM char:INTEGER
002D
002E      FOR i=1 TO 29
003E        char=ASC(MID$(name,i,1))
004D        IF char>96 AND char<123 THEN
0060          newname=newname+CHR$(char-32)
0070          changed=TRUE
0076        ELSE
007A          newname=newname+CHR$(char)
0087        ENDIF
0089      NEXT i
0094      END
PROCEDURE lower
0000      PARAM name:STRING[29]
000C      PARAM newname:STRING[29]
0018      PARAM changed:BOOLEAN
001F      DIM i:INTEGER
0026      DIM char:INTEGER
002D
002E      FOR i=1 TO 29
003E        char=ASC(MID$(name,i,1))
004D        IF char>64 AND char<91 THEN
0060          newname=newname+CHR$(char+32)
0070          changed=TRUE
0076        ELSE
007A          newname=newname+CHR$(char)
0087        ENDIF
0089      NEXT i
0094      END
PROCEDURE readfd
0000      PARAM path:BYTE
0007      PARAM attr:BYTE
000E      BASE 0
0010      TYPE registers=cc,a,b,dp:BYTE; x,y,u:INTEGER
0035      TYPE seglist=lsn(3):BYTE; nosec(2):BYTE
0050        TYPE filedes=att:BYTE; owner:INTEGER; date(5):BYTE; link:BYTE          ; sze(4):BYTE; creat(3):BYTE;
segs(48):seglist
0095      DIM regs:registers
009E      DIM fd:filedes
00A7      DIM GetStt:BYTE
00AE      DIM SS_FD:BYTE
00B5      GetStt=$8D
00BD      SS_FD=$0F
00C5      regs.a=path
00D1      regs.b=SS_FD
00DD      regs.x=ADDR(fd)
00EB      RUN syscall(GetStt,regs)
00FA      IF LAND(regs.cc,1)=1 THEN
010C        attr=-1
0114      ELSE
0118        attr=fd.att
0123      ENDIF
0125      END
```