AUSTRALIAN OS9 NEWSLETTER

Volume 8 January / February 1994 Number 1

EDITOR:	**Gordon Bentzen**	**(07) 344-3881**
SUB-EDITOR:	**Bob Devries**	**(07) 278-7209**
TREASURER:	**Jean-Pierre Jacquet**	**(07) 372-4675**
	Fax Messages	**(07) 372-8325**
LIBRARIAN:	**Rod Holden**	**(07) 200-9870**
CONSULTANT:	**Don Berrie**	**(079) 75-3537**
SUPPORT:	**Brisbane OS9 Users Group**	

CONTENTS

Editorial Material:
Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

Library Requests:
Rod Holden
53 Haig Road
LOGANLEA Qld 4131

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 8 Number 1

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

Welcome to 1994 and another year of OS-9. This edition, Volume 8 - Number 1, is the first of Volume 8 which of course is the eighth year of the National OS9 Usergoup. This group was established by owners and users of the Tandy Color Computer back in the days of what we now refer to as "the old grey case" CoCo. In those days, a few members of various Color Computer user groups became interested in Microware OS-9 Level 1 as distributed by Tandy.

Our support of 6809 OS-9 Level 1 and Level 2 continued as Tandy released new versions of the CoCo, and indeed is still supported and used by many CoCo owners.

We have at times speculated on the future of OS-9 as applicable to the personal, home users and suggested some likely alternatives to the humble Color Computer. Once Microware dropped support of the 6809 versions we looked to OS9 68000 (OSK) and a number of people in the U.S. devoted a good deal of time and effort in the development of the SUPER CoCo while others came up with new computers which would run OSK in a way that had a CoCo Level 2 look and feel.

The variations continue as newer platforms appear and a new interest is being shown in some that have been around for some time. In this edition our Bob Devries relates his first experiences with a SECAD kit.

So it seems that we, the OS-9 enthusiasts, will continue to run 6809 OS-9 in spite of it being unsupported by Microware, and we will continue the initiatives to allow ecconomical use of OSK and OS-9000.

I understand that Microware are well aware of the OS-9 Usergroups around the world and that they wish to encourage our legitimate use and interest in OS-9.

One stumbling block of course is price. Not only is the OS-9 operating system package expensive for the private user, but the hardware is also relatively expensive. It is a pity that Microware cannot offer much lower prices to NON-Commercial users on a continuous basis.
As I have noted before, I did purchase a copy of OS-9000 last year at the Chicago CoCoFest at the reduced price of US$350. Whilst this was a significant reduction from US$995, the normal price, it still worked out to be just on A$500 at the time. This special by Microware was intended to demonstrate their support of the OS-9 Usergroups and a number of copies were purchased at that function.

OS-9000 Although it is early days for OS-9000 I see this a real option. There are of course some pluses and some minuses as there are with any other operating system and hardware.

The one big advantage I see is that it runs on a standard IBM type 80386 or 80486 box. This allows me to run MS-DOS stuff which is work related, and that is something I just can't get away from, like it or not. Well I suppose the option here is to ONLY do this "stuff" at work, but it is somtimes convenient to do it at home.

I have a 386 40mhz machine with two hard drive particians (a second hard drive is not far away) which will boot MS-Dos or OS-9000 by simply hitting "M" or "O" at a boot prompt.

So it does not run on a Motorola processor as OS-9 was originally designed to do and it does suffer the hardware limitations and problems of the P.C's and clones but it does run fast and a lot of the standard 68K "C" source should compile on the OS-9000 "C" compiler.

I do not know of any other OS-9000 user in Australia, so perhaps you could contact me if you know of others.

CONTRIBUTIONS
YES! This is another appeal for material for this newsletter. How about your thoughts on where OS-9 is going? Or tell us a little about what your plans are with OS-9 and why.

Cheers, Gordon.

The National OS9 Usergroup
(07)-200-9870
300/1200/2400/9600/14400 baud.
20:00 to 21:30 HRS.(AEST)
(8N1)

Co-ordinator: Bob Devries (07)-278-7209
Sysop: Rod Holden

This is (RiBBS).... A Tandy Coco Based BBS program.
This BBS is accessible to Usergroup Members ONLY!
Feel free to look around , and test out the options.

OS9 for Ever !!!!

Hi, and welcome to all you lucky people who managed to have a holiday. We are hoping that 1994 is going to be an exciting year in the OS9 world with new software arriving throughout the year. This is your Sysop once again letting you know what type of software is available, please read on;

MORE OS9 MORE

NAME

More - a file reader/previewer with backup capabilities

SYNOPSIS

More [-options] [file1 ... fileN]

DESCRIPTION

'More' is a file reader/previewer similar to that provided with Berkeley Unix 4.3 systems. It is used to pause the file being read so that the reader may catch up. It can also be used to backup a page, move forward a page, move forward a line, move to a specific line in a file, or move to a given percentage point in a file. 'More' may be used as a filter, or the filenames may be supplied on the command line. If 'More' is reading directly from an RBF device, it will show the percentage point of where it is in the file. This is unavailable if the file is from an SCF or PIPE device. Also, 'More' may only seek to a given spot in a file if its input is from an RBF device.

Users may also use navigational commands while at the pause prompt. A file may be told to page backward, forward a page, forward a line, to go to a specific line or percentage point in the file.

The 'Q' command may be used to exit 'More' in mid-file. Also, keyboard quit and keyboard interrupt may be used for the same purpose.

Command line options:
-lnn Begin display of file at line 'nn'.
-lnn Begin display of file at percent 'nn'.
-s Use 'simple' mode
(instructions given at each prompt).

Prompt options:
SPACEBAR - Forward one page
RETURN - Forward one line
B - Backward one page
% - Go to a given percentage point in the file
L - Go to a given line in the file
! - Fork a shell command from the pause point
Q - Quit (same as QUIT or ABORT keys)

EXAMPLES
More
This filters files from standard input

More -1575 myasm.listing
This starts the display at line 575 of the file

More -%75 userlog
This seeks to the 75% point of the file and begins display there.

list sys.bulletin newuser.info ! more -s
This uses standard in as a feed, and uses the 'simple' mode for novice users (explains what to do when 'More' pauses).

BUGS
The backup algorithm sometimes moves back farther than directed. This seems to be the case primarily when the current file point is less that the size of the backup buffer (i.e. closer to the beginning of file).

The line count can occasionally get confused. It will usually hide the line count unless it 'knows' for sure where it is. After a seek operation, it cannot know. A seek to line 1 the 0 percentage point will usually serve to resynchronize the line count.

A number of other features could be added:
o - Wildcards on the command line
o - Pattern matching for start point
(both internal and command line)
o - Display of current filename (if known)
at pause prompt, on request.
o - Skipping to next or previous Nth files

This version of 'More' is being supplied on an AS-IS basis. That is, it is being uploaded lest it die in obscurity in some subdirectory. Support or maintenance is neither implied nor likely.

AUTHOR
Peter W. Lyall, Jr. 1040 Stern Lane Oxnard, CA 93035

Yes as you can see by the baud rates that I am now running a 14400 modem for those people who are lucky enough to have one. The software for 68000 and OSK is now on the BBS, but it is suggested strongly that you send disks through the mail unless you are rich to pay your phone bill at 2400 baud. My system now has 1 x 30meg, 1 x 60meg, 2 x 5 1/4 80 track, 1 x 5 1/4 40 track and 1 x 3 1/2 80 track, so there should not be a problem meeting your media requirements. See you in the bit stream, Happy CoCoing.

Sysop
Rod Holden

CLS - Revisited
by Bob Devries

Well, since I got my SECAD OSK computer running, I have been learning about the TERMCAP library. Thanks to Bob van der Poel and his series in 'The OS9 Underground', and a fairly well written example in the OSK C compiler manual, I have prepared yet another version of CLS.

When the original article about CLS appeared some time ago, Don Berrie and myself were just learning how to cope with the Atari version of OSK. We decided that we couldn't live without a CLS command, so we wrote one in assembler, because that would make the smallest binary file. Now while that is certainly true, sending a chr$(12) to the screen, is NOT PORTABLE, and is frowned upon in the OSK world.

Now we have a concept called 'the Termcap library'. There is a file called termcap in the /dd/SYS directory, which lists all of the character strings which are needed to manipulate the screen. The file doesn't only have the capabilities of MY computer, but also a variety of other computers, including the Coco3 screen (80*24). So now I can write a programme in C which will do a clearscreen on ANY computer screen, even terminals connected to my computer via the serial port(s).

There is one drawback, however. The code gets to be a LOT larger. While this was a serious consideration in the Colour Computer version, it is of less concern when we're dealing with a computer with 4MB of RAM.

So here's the code for CLS mk III:

```
#include <stdio.h>
#include <termcap.h>

#define TCAPSLEN 400

extern char *getenv();

char tcapbuf[TCAPSLEN];  /* buffer for extracted termcap strings */

char  PC_;
char  *BC;
char  *UP;
short  ospeed;
char *CL,    /* Clear screen character */
    *CM,
    *CE,
    *SO,
    *SE,
    *HO;    /* Home cursor character */

/* function to write one character */
int tputc(c)
char c;
{
    return write(1, &c, 1);
}

/* function to write a terminal control string */
putpad(str)
char *str;
{
    tputs(str, 1, tputc);
}

main()
{
    register char *term_type, *temp;
    auto char tcbuf[1024];    /* buffer for tgetent */
    auto char *ptr;

    /* find out if TERM variable has been set */
    if ((term_type = getenv("TERM")) == NULL) {
        fprintf(stderr, "Environment variable TERM not defined!\n");
        exit(1);
    }

    /* find the terminal type in termcap file */
    if (tgetent(tcbuf, term_type) <= 0) {
        fprintf(stderr,"Unknown terminal type '%s'!\n");
        exit(1);
    }

     ptr = tcapbuf;
```

```
     if (temp = tgetstr("PC", &ptr)) PC_ = *temp;   /*  get pad char
*/
     CL = tgetstr("cl", &ptr);  /* get cls char */
     HO = tgetstr("ho", &ptr);  /* get home cursor */

     putpad(HO);  /* send home cursor, just in case cl doesn't */
     putpad(CL);  /* send cls character */
}
/* EOF */
```

Well, compare all that with our earlier version, and see the major differences. Remember, however, that this one will work unconditionally, while sending a chr$(12) or whatever, MAY not. By the way, a termcap library IS available for OS9/6809. Just ask your friendly PD librarian, Rod Holden.

Bob Devries

CLUSTER'S LAST STAND
By Matthew Thompson

The following information was determined from investigations by Brian White in 1990 and my own disassembly of RBFman. So it is not just idle speculation. I am fairly confident that it is accurate and, dare I say it, authoritative.

There seems to be much confusion about the purpose and use of clustering in the OS-9 file system. The problem is largely due to the fact that the Format command, as shipped with OS-9 L2 for the CoCo 3, had the cluster support removed. Thus nobody could format with a cluster size greater than 1, and so nobody really cared.

OS-9 uses 3 bytes for the LSN number, which allows for devices with 16 million sectors, or 4 gigabytes, to be used with the OS-9 file system. No matter what the cluster size is, an LSN is always 256 bytes. Clustering does not affect the size of LSNs in any way. They are not scaled up or down or multiplied by anything. Clustering only affects how LSNs are allocated, not how big they are. So the largest drive you can use is 4 gigabytes no matter what the cluster size is.

However, one of the limitations of OS-9 is the size of the allocation bitmap. It is limited to 64K bytes because the definition of LSN0 only specifies two bytes for the number of bytes in bitmap, so the 64K limit is fairly carved in stone. But if you do the math, you get 64K * 8 sectors/byte = about 134 megs. (Technically, the bitmap is limited to 63.75K because RBFman uses a single byte internally for the bitmap sector number, and LSN0 is already used. Incidentally, the 120 megabyte limit of certain other hard drive systems is because of some technical limitation of older ST412 drives, or something like that.) That used to be adequate, but now SCSI drives with capacities of 170, 340 or more megabytes are not uncommon. So how can a file system that supports 4 gigabytes get around a bitmap limited to 134 meg? Easy, use clustering!

All clustering does is change the number of sectors represented by each bit in the allocation bitmap. Thus at a cluster size of, say 4, each bit in the bitmap represents 4 sectors, or 1024 bytes. But the LSN numbers in the file descriptor and DD.TOT are still the same old 256 sectors they always were. However, now all files must be allocated on a cluter boundary. So if the cluster size was 4, the file descriptor sector would have to occur on an LSN ending in either $0, $4, $8 or $C. Also, the first 3 sectors of the file itself would have to immediately follow the file descriptor, so they would be part of the same cluster of sectors. Plus, the segment allocation size is affected by the cluster size. SAS gets rounded up to a cluster boundary. So if you have a cluster size of 4, and SAS is set to 10, the effective SAS will be 12 as it's the next highest multiple of 4.

On the downside of clustering, there could be sectors allocated but unused because a file does not completely use a cluster. Say that a file is only 75 bytes long, and the cluster size is 4. So you have 1 sector for the file descriptor, followed immediately by 1 sector with the file, and followed by 2 sectors which are technically tagged as in use because of the cluster size, but which actually contain nothing. Such wasted space is the bane of all file systems the world over, which must balance between the efficient use of all storage space and the ease of searching and allocation files. So the higher the cluster

size, the greater the amount of space wasted, on average.

One benefit of clustering is that it reduces the size of the allocation bitmap, and thus reduces the time spent on long free space searches on full disks. Actually, a word on free space searches is in order. Some people seem to think that RBF begins a free space search from the beginning of the bitmap each time. Not so. It always remembers where it left off for each drive (in V.MapSct). BUT, if your disk is so full that there are no more free blocks at least as big as your SAS, then RBF has to scan the whole map for the next largest blob of free space. The solution is to use a smaller SAS, although admittedly this may increase the fragmentation of files and lead to more Error 217's. It is a question of choosing the lesser of two evils. Use the stock 'free' command to find the size of the largest free blob. You can also use Sacia with up to a 3.75K buffer if you are losing serial characters waiting for a free space search.

Another question is, how well is clustering handled by OS-9 and by other utilities? According to Brian White's tests, clustering is handled perfectly by the stock RBFman shipped with OS-9 L2 for the CoCo. Recently, Mike Guzzi found that the new RBFMan v30, which supports undeleting, has a bug whereby it does not properly deallocate all of the sectors when a file is deleted and the cluster size is greater than 1. While this does not corrupt the file system, it does end up wasting space. At this time I don't know the cause of the bug, but it might have something to do with the fact that the file descriptor shares the cluster with the first (and possible more) sectors of the file. Since file deletion is a system call done by RBFman, the Del command can't be the problem as it calls RBFman to do the dirty work.

As for utilities, there are problems with a number of them. The Burke and Burke hard disk utilities state in the manual that they do not support a cluster size greater than one. So you can't use Repack unless maybe they do a patch for it. Tim Kientzle's replacement Free command doesn't quite get the math right, as it seems to scale DD.TOT by the cluster size. The patched version of Ded that displays which sectors are represented when you are editing the bitmap doesn't take clustering into account. On the other hand, the stock Dcheck and Free commands get it right, so you don't have to worry about them. Dcheck does have one little bug, according to Brian White, whereby it doesn't check the last byte of the allocation bitmap. But this is even when the cluster size is 1.

Another way you can use a large hard drive is with partitioning, where one drive can be split up into several smaller logical drives. This way, you can keep the cluster size at one 1 on each drive. Since RBFman doesn't support partitioning, it has to be a function of the device driver to do it. The SCSI System now includes support for partitioning if you want it.

Recently, a revised format command, Mformat, has been made available to format any disk with a cluster size beyond 1. So you can now use clustering on any OS-9 disk. Also, the SCSI System comes with its own formatter which supports higher cluster sizes.

A related question to clustering is, how would it be if RBFMan handled 512 byte sectors natively? Under OS-K 2.4, RBFman can configure things so that an LSN really is 512 bytes, and file descriptors and LSN0 are also 512 bytes long, etc. Well, technically it's possible, but not practical, under OS-9 6809, for a number of reasons. Keep in mind that I actually tried to rewrite RBFman myself to do this, and after several months realized that it wasn't worth it.

First, it would require a major rewrite of RBFMan, and would expand its size considerably, gobbling up precious system space in Level 2. Second, every open path would require a 512 byte buffer, gobbling even more system space. Third, 256 byte sectors lend themselves nicely to quantities that fit in one byte, a fact which RBF uses extensively. Going to 512 would mean needing two bytes, and this would mean changing drive table and path extension definitions around. And if the value was in A or B you couldn't just use D because the other accumulator usually had something significant in it as well. Finally, many of your favorite OS-9 utilities just aren't ready for the shock of 512 byte sectors, and would need to be rewritten (ie 'Ded'). So while going to 512 natively under OS-K wasn't too hard, it would mean a lot of headaches under OS-9 L2.

Any driver which support 512-byte sectors simply uses various tricks to make the drive look like 256 to RBFMan. So you have one block on the device holding two logical OS-9 sectors. While it is sort of a kluge, but it seems to do the trick! And you don't have to modify any utilities or other software to handle it.

The SECAD AS-68K
"first experiences" - by Bob Devries

I bought a SECAD AS68K computer as a kit a few months ago, and have, over the Christmas break, been able to get it together, and to work successfully. While most of my success was probably because I am an electronics technician, I did have a lot of help from Mr Jim Adamthwaite, one of the partners of the SECAD Systems company. Without his careful, though sometimes wordy, explanations, I doubt if I could have got it 'just right'.

The kit I purchased, consisted of the PCB, with the ROMS, and PAL chips (that's Programmable Array Logic), and OS9/68000 Professional. The PCB has some 95 Integrated Circuits on it, 295 components all up. It took some time to place and solder all the components, and then check to see everything was OK.

The CPU is a 68000, running at 10Mhz, with a 68450 DMAC (Direct Memory Access Controller), a 68681 DUART (Dual Universal Receiver Transmitter) for serial ports, and a 68B21 PIA (Peripheral Interface Adaptor) for the keyboard. The computer uses 'standard' IBM XT bus sockets, and a IBM keyboard. Standard IBM I/O cards are used for floppy disks, hard disks, and screen. I am currently using a CGA card for my screen, but I hope to upgrade to EGA soon. My system at this moment has the following (subject to change of course):

CGA color screen (using Thomson EGA monitor)
WD TM-262 20MB hard disk with DTC 5150 controller
Digitor (from Disk Smith) multi IO card for floppy, serial and printer.
2 * 720K 3.5" floppy drives
4 Megabytes of RAM (the maximum the PCB allows)

OS9/68000 came on 3 * 3.5" 'universal format' floppies, and the manuals (two) are A5 size, approx 60mm thick each.

Putting the board together was fairly straight-forward, although there is no construction manual. I only had two problems, one was because I assumed that the lowest 2MB of RAM would be in the lowest numbered sockets. Of course, I was wrong, and I got a buss error, and the system just stopped (or rather, HALTed). The other problem was a wrong capacitor type connected to the timing crystal for the IBM bus, which caused 'glitches' all over my screen.

Once running, getting the software installed was no real problem, except that I needed to compile some of the drivers, since they didn't come in binary form, only in assembler form. Another call to the patient Jim Adamthwaite. I was even able to modify an existing device descriptor to be able to read Colour Computer OS9 format 3.5" disks.

Of course, learning to use OS9/68000, or rather UN-learning OS9/6809, will take some time. As Gordon will be able to tell you, learning the new OS has a rather steep learning curve, and his OS9000 is worse, because there are some MAJOR differences there. For the moment, I have to be content with using uMacs, the screen editor supplied, since I don't yet have a copy of VED/68000. I have a demo version, and it feels just like the OS9/6809 version. I'm using uMacs now, because my Colour computer's disk controller died a few days ago, and it will take some time to fix (it's a Disto SCII).

Lots of the standard utilities work the same, or at least have enough similarity to OS9/6809 for me to be quite at home behind the SECAD's keyboard. In fact, some of the utilities have options which would be great in the 6809 version, but are probably not there because of the memory restraints. Most of the utilities in OSK are written in C originally, and so are longer than their assembler counterparts, where the same ones for os9/6809 are in assembler, and so are smaller.

So now I have an OSK computer, with 4 serial ports, a printer port, two disk drives, one hard drive, one internal modem, and a screen which will do 640*200*2 or 320*200*4, a mouse, an 84 key keyboard.

Software, you ask? Yes, well, ahem. Well, I'll be starting to write some soon. Of course there are quite a number of PD archives available, including some 14.5 MB from the EFFO group (European Forum For OS9), which include a C compiler, a Forth compiler, and lots of utilities. These files are available from our PD library, but you MUST have 720K disks, or they won't fit. In fact a couple won't fit on even that size, so they'll have to be split up. There are files PD1.lzh to PD9.lzh and FORUM01.lzh to FORUM23.lzh.

Well that's all for now. I'll tell some more of my experiences next time. If anyone is interested in buying this kit (or fully built-up) computer, contact Jim Adamthwaite at:

SECAD SYSTEMS
66 Albert Street Ph: (03) 380 9036
Brunswick East Vic 3057 Int'l +61 3 380 9036

Bootsplit
by Bob Devries

Having recently bought a copy of OS9/68000, I found I needed to split a bootfile, to remove a couple of modules, and add new ones. So, of course, I'd use the bootsplit utility, just like in OS9/6809 right? Wrong!!! It doesn't exist. So, what to do.... Write one! The code for this one is really not very difficult. It is merely necessary to open the file, read enough data from it to fill a structure (from the module.h file), so as to find out where the module name, and length are to be found. Well, as you can see, it's not too simple, since all sorts of checks for validity need to be done, and the module in question must be able to be loaded into memory from the merged bootfile.

Of course, bootsplit can be used on ANY merged file of modules. Although, unlike OS9/6809, these appear to be fewer in OS9/68000. Anyway, here's the C source code:

```
/* Bootsplit for OS9/68000   */
/* by Bob Devries. (c) 1993  */
/* may be freely distributed */

#include <stdio.h>
#ifdef OSK
#include <module.h>
#else
#include "module.h"
#endif
#include <errno.h>

main(argc,argv)
int argc;
char *argv[];
{
        char modfile[33];
        char *malloc(),*buffer;
        long modpos = 0L;
        FILE *fopen(), *ifp, *ofp;
        struct modhcom module;
        int nread;

        if ((argc < 2) || (argv[1][0] == '-')) {
                usage();
                exit(0);
        }

        if((ifp=fopen(argv[1],"r")) == NULL)
                exit(_errmsg(errno,"Can't open %s\n",argv[1]));
        for(;;) {
                if((nread=fread(&module,sizeof(module),1,ifp))    ==
NULL)
                        exit(_errmsg(errno,"Done.\n"));
                fseek(ifp,modpos,0);
                buffer = malloc(module._msize);
                if(buffer == NULL)
                        exit(_errmsg(1,"Not enough memory.\n"));
                fread(buffer,module._msize,1,ifp);
                modpos += module._msize;
                strcpy(modfile,buffer + module._mname);
```

```
                if((ofp=fopen(modfile,"w")) == NULL)
                        exit(_errmsg(errno,"Can't                open
%s.\n",modfile));
                printf("%s\n",modfile);
                fwrite(buffer,module._msize,1,ofp);
                fclose(ofp);
                free(buffer);
        }
        fclose(ifp);
}

usage()
{
        fprintf(stderr,"Usage: %s <filename>\n",_prgname());
        fprintf(stderr,"Splits merged module to separate files.\n");
}
```

NOTE: For this programme to compile under OS9/6809, you'll need to use the kreider library, and use a different version of the module.h header file. Here's the code for this file:

```
/* alternate version of module.h for OS9/68000 compatibility */

struct modhcom {
        unsigned        _msync,         /* sync bytes ($87cd) */
                        _msize,         /* module size */
                        _mname;         /* offset to module name */
        char            _mtylan,        /* type & language */
                        _mattrev,       /* attributes & revision */
                        _mparity;       /* header parity */
};
```

ED NOTE :-
We do hope that you find something useful in these pages and urge you to make a contribution of material suitable for inclusion in the next newsletter.
Your masterpiece should reach the Editor by the end of February.

Until next month, HAPPY COMPUTING !