AUSTRALIAN OS9 NEWSLETTER

sides 'No. of cylinders' (in decimal) :Interleave value: (in decimal) @FREE Syntax: Free [devname] Usage : Displays number of free sectors on a device @GFX Syntax: RUN GFX(<funct><args>) Usage : Graphics interface package for BASIC09 to do compatible VDG graphics commands @GFX2 Syntax: RUN GFX2([path]<funct><args>) Usage : Graphics interface package for BASIC09 to handle ...ax: none Usage : ...graphics/ window... ...on-line help to ... ...topics @IDENT... ...rmation from OS... ...s = use single lin... ...ecution directory... ...device @INKE... ...utine to input a ... ...abort to the proc... ...nk to a memory... ...ntents of text files @LOAD Syntax: Load <pathname> [-] Usage : Loads modules into memory @MAKDIR Syntax: Makdir <pathname> Usage : Creates a new directory file @MDIR Syntax: Mdir [e] Usage : Displays the present memory module directory Opts : e = print extended module directory @MERGE Syntax: Merge <path> ...ndard output @MFREE Synt... ...RAM memory @MODPATCH... ...a module in memory from c... ...warnings -c = compare modul... ...odname = link to module C o... ...V = verify module M = ma... ...Montype [opt] Usage : Set mo... ...monitor m = monochrome m... ...sage : Creates and links an OS... ...ROCS Syntax: Procs [e] Usage... ...em Opts : e = display all proc... ...: Prints the current data directory path @PXD Syntax: Pxd Usage : Prints the current execution directory path @RENAME Syntax: Rename <filename> <new filename> Usage : Gives the file or directory a new name @RUNB Syntax: Runb <i-code module> Usage : BASIC09 run time package @SETIME Syntax: Setime [yy/mm... ...ETPR Syntax: ...s to num @... ...ter @TMODE Syntax: Tmode [pathname] [params] Usage : Displays or changes the operating parameters of the terminal @TUNEPORT Tuneport <t1 or /p> [value] Adjust the baud value for the serial port @UNLINK Syntax: Unlink <modname> Usage : Unlinks module(s) from memory @WCREATE Syntax:

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-1284
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
	Fax Messages	(07) 372-8325
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 November 1991 Number 10

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 5 Number 10

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

As you can see we are still producing the newsletter, and in fact we have changed the format to give you more information for your money. With smaller margins all round, we can fit more words on each page. Of course, this means we have to think up more to go into it.

One of the hardest things that we have to do is to try to provide articles of interest to our readers. It would be extremely helpful to us, if you could provide us with some ideas as to the type of article that YOU would like to read. To do this, simply drop us a line, and we will take it from there.

We would, for example, like to publish articles which include source code, but many of the public domain submissions that we receive, or that we get from overseas sources, only contain executable code. This almost always precludes the use of those submissions as material for the Newsletter. We figure that it is better to distribute that material on disks via our Public Domain library system.

This month we feature a Basic09 programme by Bob Devries which will read an RSDos 'tokenised' file (i.e. one saved normally, not in ASCII), and print it to standard out in ascii. This means, that if you want to convert such a programme to Basic09, all you need do is read it from your RSDos disk (using the programme RSDos) and run it through this programme, with its output re-directed to a file. This will save having to go back to RSDos and convert the file to ASCII.

We also have more text from the BitNet system with discussions about 'Windows' on OSK machines. It seems that there are quite a lot of different ideas about how that should be implimented.

For those who have a lot of back-issues of 'The Rainbow', we have continued the listing of the index of all the OS9 articles which appeared, along with their authors, and page numbers.

We have some news items from American OS9 users, and what they have to put up with over there. It makes some very interesting reading.

Well as you can guess I am back from Europe and the holiday is over. I must say that it is good to be back home again to things familar. I will never complain again about the cost of living in Australia again but the experience in U.K. and Europe is one not to missed.

We do hope that you find something of interest in this edition and remind you all that we will be happy to include any articles which you may wish to share with other members.

Regards, Gordon

Read RSDos BASIC tokenised files
by Bob Devries

Here is a programme which will read a RSDos BASIC tokenised file and print it out as ASCII text. You must of course have some way of moving the file from a RSDos disk to OS9, but there is already a utility to do this called 'RSDos'. It is in our PD library.

This programme expects the filename on the command line, and prints the ASCII to STDOUT, so you can redirect it to a file, or the printer. If you don't give it a name, it prints a usage message. This is donr by setting up an ON ERROR GOTO line first up, and then using the PARAM line, which reads in the variables from the command line. If none exist, an error 56 would occur, but as it is trapped, the programme quits cleanly.

After opening the file, the programme jumps over the first five bytes, and reads the next two to get the first line number. It then enters a loop to keep reading characters until a zero value is read, which is the end of line character, and prints a carriage return. After skipping two bytes (the address pointer to the next line), it reads two bytes for the next line number. This goes on until three zero bytes are read in a row, which means the end of the programme has been reached. The file is then closed and the programme ends.

Here is the source.

```
PROCEDURE translate
 0000       ON ERROR GOTO 1000
 0006       PARAM file:STRING[100]
 0012       DIM token:BYTE
 0019       DIM command:STRING[20]
 0025       DIM path:INTEGER
 002C       DIM x:INTEGER
 0033       DIM flag,ernum:INTEGER
 003E       DIM line:INTEGER
 0045       flag=0
 004C       OPEN #path,file:READ
 0058       FOR x=1 TO 5
 0068         GET #path,token
 0072       NEXT x
 007D       GET #path,token
 0087       line=token*256
 0093       GET #path,token
 009D       line=line+token
 00A9       PRINT line; " ";
 00B3       WHILE NOT(EOF(#path)) DO
 00BE         GET #path,token
 00C8         IF flag=1 THEN
 00D4           line=token*256
 00E0           GET #path,token
 00EA           line=line+token
 00F6           IF line=0 THEN
 0102             END
 0104           ENDIF
 0106           GET #path,token
 0110           line=token*256
 011C           GET #path,token
 0126           line=line+token
 0132           PRINT line; " ";
 013C           flag=0
 0143           GET #path,token
 014D         ENDIF
 014F         IF token<>0 THEN
 015B           flag=0
 0162           IF token>127 THEN
 016E             token=token-127
 0179             IF token=128 THEN
 0185               RESTORE 1020
 018A               GET #path,token
 0194               token=token-127
 019F             ELSE
 01A3               RESTORE 1010
 01A8             ENDIF
 01AA             FOR x=1 TO token
 01BB               READ command
 01C0             NEXT x
 01CB             PRINT command;
 01D1           ELSE
 01D5             PRINT CHR$(token);
 01DC           ENDIF
 01DE         ELSE
 01E2           PRINT
 01E4           flag=1
 01EB         ENDIF
 01ED       ENDWHILE
 01F1       CLOSE #path
 01F7       END
 01F9 1000  ernum=ERR
 0202       IF ernum=56 THEN
 020E         PRINT "usage:translate <filename>"
 022C         PRINT "      converts  RSDos basic tokenized
programme to ASCII"
 0267         END
 0269       ENDIF
 026B       PRINT "OS9 error "; ernum; " has occured!"
 028D       END

 028F 1010  DATA "FOR","GO","REM","'","ELSE","IF","DATA","PRINT","ON","INPUT"
 02D3       DATA "END","NEXT","DIM","READ","RUN","RESTORE","RETURN","STOP"
 0311       DATA "POKE","CONT","LIST","CLEAR","NEW","CLOAD","CSAVE","OPEN"
 034F       DATA "CLOSE","LLIST","SET","RESET","CLS","MOTOR","SOUND","AUDIO"
```

```
038F      DATA "EXEC","SKIPF","TAB(","TO","SUB","THEN","NOT","STEP","OFF"
03CE      DATA "+","-","*","/","^","AND","OR",">","=","<"
03FD      (* commands for ECB follow *)
041A      DATA "DEL","EDIT","TRON","TROFF","DEF","LET","LINE","PCLS"
0454      DATA "PSET","PRESET","SCREEN","PCLEAR","COLOR","CIRCLE"
048B      DATA "PAINT","GET","PUT","DRAW","PCOPY","PMODE","PLAY","DLOAD"
04C9      DATA "RENUM","FN","USING"
04E2      (* commands for DECB *)
04F9      DATA "DIR","DRIVE","FIELD","FILES","KILL","LOAD","LSET","MERGE"
0538      DATA "RENAME","RSET","SAVE","WRITE","VERIFY","UNLOAD"
056D      DATA "DSKINI","BACKUP","COPY","DSKI$","DSKO$","DOS"
05A0      (* commands for SECB (CC3) *)
05BD      DATA "WIDTH","PALETTE","HSCREEN","LPOKE","HCLS","HCOLOR"
05F5      DATA "HPAINT","HCIRCLE","HLINE","HGET","HPUT","HBUFF"
062A      DATA "HPRINT","ERR","BRK","LOCATE","HSTAT","HSET","HRESET"
0664      DATA "HDRAW","CMP","RGB","ATTR"
0683      (* function commands CB *)
069D 1020 DATA "SGN","INT","ABS","USR","RND","SIN","PEEK","LEN","STR$"
06DC      DATA "VAL","ASC","CHR$","EOF","JOYSTK","LEFT$","RIGHT$","MID$"
071A      DATA "POINT","INKEY$","MEM"
0735      (* function command for ECB *)
0753      DATA "ATN","COS","TAN","EXP","FIX","LOG","POS","SQR","HEX$"
078E      DATA "VARPTR","INSTR","TIMER","PPOINT","STRING$"
07BE      (* function commands for DECB *)
07DE      DATA "CVN","FREE","LOC","LOF","MKN$","AS"
0807      (* function commands for SECB (CC3) *)
082D      DATA "" \(* empty one for CC3 bug *)
084F      DATA "LPEEK","BUTTON","HPOINT","ERNO","ERLIN"
```

oooooooooo0000000000oooooooooo

A C Tutorial
Chapter 4 - Assignment & Logical compares
Continued...

MORE COMPARES

The compares in the second group are a bit more involved. Starting with the first compare, we find a rather strange looking set of conditions in the parentheses. To understand this we must understand just what a "true" or "false" is in the C language. A "false" is defined as a value of zero, and "true" is defined as a non-zero value. Any integer or char type of variable can be used for the result of a true/false test, or the result can be an implied integer or char. Look at the first compare of the second group of compare statements. The expression "r != s" will evaluate as a "true" since "r" was set to 0.0 above, so the result will be a non-zero value, probably 1. Even though the two variables that are compared are "float" variables, the result will be of type "integer". There is no explicit variable to which it will be assigned so the result of the compare is an implied integer. Finally the resulting number, 1 in this case, is assigned to the integer variable "x".

If double equal signs were used, the phantom value, namely 1, would be compared to the value of "x", but since the single equal sign is used, the value 1 is simply assigned to "x", as though the statement were not in parentheses. Finally, since the result of the assignment in the parentheses was non-zero, the entire expression is evaluated as "true", and "z" is assigned the value of 1000. Thus we accomplished two things in this statement, we assigned "x" a new value, probably 1, and we assigned "z" the value of 1000. We covered a lot in this statement so you may wish to review it before going on. The important things to remember are the values that define "true" and "false", and the fact that several things can be assigned in a conditional statement. The value assigned to "x" was probably a 1 but different compilers may assign a different value as long as it is non-zero.

The next example should help clear up some of the above in your mind. In this example, "x" is assigned the value of "y", and since the result is 11, the condition is non-zero, which is true, and the variable "z" is therefore assigned 222. The third example, in the second group, compares "x" to zero. If the result is true, meaning that if "x" is not zero, then "z" is assigned the

value of 333, which it will be. The last example in this group illustrates the same concept, since the result will be true if "x" is non-zero. The compare to zero is not actually needed and the result of the compare is true. The third and fourth examples of this group are therefore identical.

ADDITIONAL COMPARE CONCEPTS

The third group of compares will introduce some additional concepts, namely the logical "AND" and the logical "OR". We assign the value of 77 to the three integer variables simply to get started again with some defined values. The first compare of the third group contains the new control "&&", which is the logical "AND". The entire statement reads, if "x" equals "y" AND if "x" equals 77 then the result is "true". Since this is true, the variable z is set equal to 33. The next compare in this group introduces the " " operator which is the "OR". The statement reads, if "x" is greater than "y" OR if "z" is greater than 12 then the result is true. Since "z" is greater than 12, it doesn't matter if "x" is greater than "y" or not, because only one of the two conditions must be true for the result to be true. The result is true, so therefore "z" will be assigned the value of 22.

LOGICAL EVALUATION

When a compound expression is evaluated, the evaluation proceeds from left to right and as soon as the result of the outcome is assured, evaluation stops. Namely, in the case of an "AND" evaluation, when one of the terms evaluates to "false", evaluation is discontinued because additional true terms cannot make the result ever become "true". In the case of an "OR" evaluation, if any of the terms is found to be "true", evaluation stops because it will be impossible for additional terms to cause the result to be "false". In the case of additionally nested terms, the above rules will be applied to each of the nested levels.

PRECEDENCE OF OPERATORS

The question will come up concerning the precedence of operators. Which operators are evaluated first and which last? There are many rules about this topic, which your compiler will define completely, but I would suggest that you don't worry about it at this point. Instead, use lots of parentheses to group variables, constants, and operators in a way meaningful to you. Parentheses always have the highest priority and will remove any question of which operations will be done first in any particular statements. Going on to the next example in group three, we find three simple variables used in the conditional part of the compare. Since all three are non-zero, all three are "true", and therefore the "AND" of the three variables are true, leading to the result being "true", and "z" being assigned the value of 11. Note that since the variables, "r", "s", and "t" are "float" type variables, they could not be used this way, but they could each be compared to zero and the same type of expression could be used.

Continuing on to the fourth example of the third group we find three assignment statements in the compare part of the "if" statement. If you understood the above discussion, you should have no difficulty understanding that the three variables are assigned their respective new values, and the result of all three are non-zero, leading to a resulting value of "TRUE".

THIS IS A TRICK, BE CAREFUL

The last example of the third group contains a bit of a trick, but since we have covered it above, it is nothing new to you. Notice that the first part of the compare evaluates to "FALSE". The remaining parts of the compare are not evaluated, because it is an "AND" and it will definitely be resolved as a "FALSE" because the first term is false. If the program was dependent on the value of "y" being set to 3 in the next part of the compare, it will fail because evaluation will cease following the "FALSE" found in the first term. Likewise, "z" will not be set to 4, and the variable "r" will not be changed.

POTENTIAL PROBLEM AREAS

The last group of compares illustrate three possibilities for getting into a bit of trouble. All three have the common result that "z" will not get set to the desired value, but for different reasons. In the case of the first one, the compare evaluates as "true", but the semicolon following the second parentheses terminates the "if" clause, and the assignment statement involving "z" is always executed as the next statement. The "if" therefore has no effect because of the misplaced semicolon. The second statement is much more straightforward because "x" will always be equal to itself, therefore the inequality will never be true, and the entire statement will never do a thing, but is wasted effort. The last statement will always assign 0 to "x" and the compare will therefore always be "false", never executing the conditional part of the "if" statement. The conditional statement is extremely important and must be thoroughly understood to write efficient C programs. If any part of this discussion is unclear in your mind, restudy it until you are confident that you understand it thoroughly before proceeding onward.

THE CRYPTIC PART OF C

There are three constructs used in C that make no sense at all when first encountered because they are not

intuitive, but they greatly increase the efficiency of the compiled code and are used extensively by experienced C programmers. You should therefore be exposed to them and learn to use them because they will appear in most, if not all, of the programs you see in the publications. Load and examine the file named CRYPTIC.C for examples of the three new constructs. In this program, some variables are defined and initialized in the same statements for use below. The first executable statement simply adds 1 to the value of "x", and should come as no surprise to you. The next two statements also add one to the value of "x", but it is not intuitive that this is what happens. It is simply by definition that this is true. Therefore, by definition of the C language, a double plus sign either before or after a variable increments that variable by 1.

Additionally, if the plus signs are before the variable, the variable is incremented before it is used, and if the plus signs are after the variable, the variable is used, then incremented. In the next statement, the value of "y" is assigned to the variable "z", then "y" is incremented because the plus signs are after the variable "y". In the last statement of the incrementing group of example statements, the value of "y" is incremented then its value is assigned to the variable "z". The next group of statements illustrate decrementing a variable by one. The definition works exactly the same way for decrementing as it does for incrementing. If the minus signs are before the variable, the variable is decremented, then used, and if the minus signs are after the variable, the variable is used, then decremented.

THE CRYPTIC ARITHMETIC OPERATOR

Another useful but cryptic operator is the arithmetic operator. This operator is used to modify any variable by some constant value. The first statement of the "arithmetic operator" group of statements simply adds 12 to the value of the variable "a". The second statement does the same, but once again, it is not intuitive that they are the same. Any of the four basic functions of arithmetic, "+", "-", "*", or "/", can be handled in this way, by putting the function desired in front of the equal sign and eliminating the second reference to the variable name. It should be noted that the expression on the right side of the arithmetic operator can be any valid expression, the examples are kept simple for your introduction to this new operator. Just like the incrementing and decrementing operators, the arithmetic operator is used extensively by experienced C programmers and it would pay you well to understand it.

THE CONDITIONAL EXPRESSION

The conditional expression is just as cryptic as the last two, but once again it can be very useful so it would pay you to understand it. It consists of three expressions within parentheses separated by a question mark and a colon. The expression prior to the question mark is evaluated to determine if it is "true" or "false". If it is true, the expression between the question mark and the colon is evaluated, and if it is not true, the expression following the colon is evaluated. The result of the evaluation is used for the assignment. The final result is identical to that of an "if" statement with an "else" clause. This is illustrated by the second example in this group. The conditional expression has the added advantage of more compact code that will compile to fewer machine instructions in the final program. The final two lines of this example program are given to illustrate a very compact way to assign the greater of two variables "a" or "b" to "c", and to assign the lessor of the same two variables to "c". Notice how efficient the code is in these two examples.

TO BE CRYPTIC OR NOT TO BE CRYPTIC

Several students of C have stated that they didn't like these three cryptic constructs and that they would simply never use them. This would be fine if they never have to read anybody else's program, or use any other programs within their own. I have found many functions that I wished to use within a program but needed a small modification to use it, requiring me to understand another person's code. It would therefore be to your advantage to learn these new constructs, and use them. They will be used in the remainder of this tutorial, so you will be constantly exposed to them. This has been a long chapter but it contained important material to get you started in using C. In the next chapter, we will go on to the building blocks of C, the functions. At that point, you will have enough of the basic materials to allow you to begin writing meaningful programs.

PROGRAMMING EXERCISES

1. Write a program that will count from 1 to 12 and print the count, and its square, for each count.

1	1
2	4
3	9 etc.

2. Write a program that counts from 1 to 12 and prints the count and its inversion to 5 decimal places for each count. This will require a floating point number.

1	1.00000
2	.50000
3	.33333
4	.25000 etc.

3. Write a program that will count from 1 to 100 and print only those values between 32 and 39, one to a line.

OSK Windows

Tim Kientzle, a regular writer from the CoCo Mailing List, has posted the following (somewhat lengthy) message, as a precis of a number of discussions about windowing systems for the new (and existing) OSK machines. Unfortunately, it assumes a knowlege of some of the prior discussions that have taken place on the List. It is still worthwhile reading, however, as it raises some interesting possibilities about uses of more sophisticated windowing systems - Ed.

Wheeee! Boy this discussion took off. Ed Gow's basic point seems to be to Keep It Simple, which is something I agree with. It does seem that there are different ideas of "simple" around here, though. <grin> I feel like I've pretty well outlined my ideas, so I'll try to keep my responses here as simple as possible... (though I'll doubtless fail <sigh>). I've edited this to keep it short (!), so I apologize if things are unclear.

> Ed has concerns about my comment regarding applications re-opening at the same position they were last at.

Frank Hogg's idea of having applications keep configuration files in the user's home directory was essentially what I had in mind. The biggest reason I mentioned this was to make a case for allowing the application to specify where it's window belongs on the screen, as opposed to Eddie Kunz's idea that the user should do that.

Ed asks about another detail:

> This raises the window/screen thing again. What if the window change requires a screen change?? What of the other windows on the screen??

One idea is simply to ignore requests that might require a screen change. Thus, an application could make sure it got a window on the current screen by asking for a 1x1 window with 1 color, then create the window, then ask for more. The application always has the option of closing the window and creating a new one if a new screen is acceptable.

Another idea is that if the change would force a screen change, you change the screen, then ask the other windows to re-draw. (Note: Any system that allows overlapping windows has to address the question of how to handle exposures. It seems to me that the burden of re-drawing exposed graphics windows must rest on the application. Bit-mapped backing store just takes too much memory.

Though I also feel that the window manager should be responsible for re-drawing text windows in order to keep simple programs simple.)

Greg Law points out that:

> Obviously you don't always need the max resolution with max colors.

Well put. Hopefully, if the window manager knows what the applications really _need_, it can do clever things like creating a monochrome screen if there's not enough video memory for something fancier.

Mike Knudsen comments about Eddie's DWSet idea:

> Good idea -- a small set of MM/1 window types that can be met on other platforms as well. And right, don't change em!

This is a reasonable idea, except that the "met on other platforms" requires some conservatism. For example, the VSC supports a 720-across screen, but that's an uncommon width, so your "standard" type codes would have to stop at 640-across. (Or maybe smaller?)

Ed Gow:

> Most programs can run in all screen types. The heavy-duty graphics progs will probably want their own screen anyway.

I agree, but with a minor clarification. Most heavy-duty graphics programs will want a large window, not necessarily their own screen, as Mike already pointed out. Some heavy-duty graphics progs will desperately want to limit their market <grin> by directly messing with video memory. THOSE programs must be given their own screen, so they won't damage anyone else's window. ;-)

(A good example might be an X windows server...???)

Ed Gow asks for some clarification:

> There is substantial state maintained between DWWant() calls, who has it?

The window manager creates a window description when the path is opened, and alters it as requested by the program. Once the window is actually created/displayed ("mapped" in X parlance), then some of those parameters may become unalterable.

Ed Gow points out a simplification I've been making:

> It looks as though the window calls are making screens in the description above, but they lack suficient information to really specify a screen type (border/no border, interlace, etc.).

First, those DWWant/DWNeed calls aren't creating screens or windows, but just specifying attributes that the window should have when it is created. Attributes for which the application doesn't express a preference will default.

I've omitted some window properties that some programs will want to specify. By having each call specify a path number, a code specifying the parameter, and the parameter value, we gain the flexibility of being able to easily add more codes and hence more window options at a later date. Changable parameters might include: x-resolution in pixels, y-res in pixels, x-res/y-res in chars, name/style/size of font, type of window border, number of colors, title of window, mouse pointer when in window, background color of window, etc.

Also, one note: I don't think interlace is something that the application should ever know about, much less be able to request. I think that the user _should_ be able to configure the window manager to prefer interlaced or non-interlaced screens. I don't want some application constantly selecting an interlaced screen when I would rather have it select a non-interlaced one. Or vice-versa.

Mike Knudsen is concerned that

> All this platform independence is slowing down Kevin Darling's development and even causing useful features to be omitted or implemented in complex, inefficient ways.

For the most part, the CoCo windowing design is already pretty hardware independent. There are only a few real limitations in it, such as the method of specifying palettes, and the method of specifying a window or screen type, and some less-important restrictions on networking and use of remote terminals (mouse and get/put buffer mapping). Extensions such as overlapping windows, etc, only have minor impact on the overall design. As someone mentioned, a little work in getting the basic interface right could make a big difference later on. There really are only a couple of issues.

Ed Gow comments that:

> I think that screen creation should be explicit. There can be some negotiation, but I think that it would be simplified because of the limited number of choices of screens, as opposed to the unlimited types of windows. Then the window can be made given the screen constraints.

I don't see this as a simplification. It seems simpler to me for the program author to simply think in terms of what that program needs in terms of resolution, colors, etc, and just ask for that. Pushing screen selection onto the application just seems redundant. Why should every application have to ask the window manager what kinds of screens are available and then have to search that list to figure out which type it prefers, when code to compare an application's requirements to a list of possible screen types could be written once and hidden away in the window manager?

I do see one case where the application would need to specify a screen, and that is in the rare case where the application _needs_ a quick response from the user, where it might ask that a window be created on the currently visible screen, in order to get the user's immediate attention.

I can certainly see a case for screen creation being explicit. Unfortunately, I don't see a convenient way to do this that wouldn't make either the application or the window manager more complex.

Suggestions, as always, are welcome.

Ed Gow asks

> Let WINDOW creation apply only to the current screen or a screen explicitly created by the application for the window. Most programs can run in all screen types.

True, and if the window being requested will fit, it seems clear to me that the current screen is the obvious place to put that window. But what if it doesn't fit? What you seem to be suggesting is that either the user must explicitly create a new screen, then select that screen, then somehow tell the application to open it's window there; or the application should ask the user for permission to create a new screen and do this (which seems to suggest that the application must be able to specify what screen to put a window on). As a user, the first one doesn't sound very simple. As a programmer, the second adds the burden of having to worry about screens and windows, rather than just windows. Perhaps you have some proposed interface that would make it easier than it looks right now? Or maybe I'm missing something obvious? Hmmmm....

In order to help explain my proposal and viewpoint better, here's some random thoughts:

I should briefly describe here my personal "vision" of a nice windowing environment. Some of this comes from ideas which were actually implemented (?) in some version of the OS9 L2 upgrade, so I'm certain that Kevin can implement much of this in some future version of DWindows. My "vision" involves having separate screens, selectable via function keys or mouse. Each screen can have multiple overlapping windows. I've heard that this is actually available in at least one X window manager already (tvtwm?).

For myself, I see using this to organize things by task. A screen where I've been working on an article might have a word processor with a separate preview window and an on-line dictionary. Another screen where I'm working on a program would have a couple of editor windows, a compiler, and a debugger window. A third screen is where I'm logged into Delphi, with a terminal program, an editor where I'm composing replies to forum messages, and a shell window where I'm trying to play with answers to someone's question. All the screens might have clocks on them, or maybe one clock program with a window on each screen? You get the idea. The question is how can the user (me ;-) easily access these capabilities?

The idea I've been working with is that rather than setting this up by explicitly creating screens or designating certain screens, I would simply run the program I want, which would appear on the current screen unless it required something unavailable there (like too many colors). If I wanted it on a different screen, I drag it to another screen. One of the screens in the "drag cycle" would always be a blank screen (assuming available memory) so I can set up things however I wish. If I'm starting up a graphics editor which requires more palettes than the current screen has, then I find myself looking at that graphics editor on a new screen. Which gives me the option of either dragging other windows onto this screen, or flipping back and forth. As long as basic window manipulations are fast and easy, then the user has convenient flexibility to arrange things as they wish.

Everyone seems to agree that a mechanism to create a new window should specify the resolution, colors, and other attributes that window should have. My idea essentially boils down to a method for the application to specify the things it cares about and let the window manager choose "reasonable" defaults for the rest. Most applications would only specify the dimensions of the window and the number of colors and ignore the rest. If the idea of letting the window manager select a screen and screen type based on this information seems unwieldy (I'll admit it does have some drawbacks), and you think the application should be able to control/influence the creation of new screens, please tell me (and Kevin, I suppose) how to do it. ;-)

- Tim Kientzle

oooooooooo0000000000oooooooooo

An index of Rainbow OS9 articles
compiled by Bob Devries
August '85 - December 1986

August 1985 page 236
KISSable OS9 - Cliffhangers in the micro soaps
Dale L. Puckett

August 1985 page 246
MAIL09 - The remainder of MAIL09's listings
Timothy A. Harris

September 1985 page 238
KISSable OS9 - A getting-your-feet-wet course in OS9 Pascal
Dale L. Puckett

October 1985 page 242
KISSable OS9 - OS9 gets good reception at National Computer Conference
Dale L. Puckett

November 1985 page 218
KISSable OS9 - Confessions of an enlightened spreadsheet user
Dale L. Puckett

November 1985 page 241
Learning OS9 - The utility room
Brian A. Lantz

December 1985 page 272
KISSable OS9 - A time for reflection
Dale L. Puckett

December 1985 page 258
The Utility Room - Adding more features to the LIST command, part 2
Brian A. Lantz

January 1986 page 134
Getting Started with OS9 - Boot up with a high level

oooooooooo0000000000ooooooooo

NEWS Items

The following are some examples of prices which are available to our USA collegues. It makes your mouth water, and your mind wonder as to whom is making what profits here in Australia. If you don't believe me, check for prices of similar items locally. Prices are in US dollars.

1Mx9 SIMMS (80ns) $49.77

BRAND NEW Sony 3.5" floppy drives
720K, WITH a 5.25" Mounting Kit, $29 each!!!
($25 each in quantities of 10 or more)

1.44M, $49 each!!!
($39 each in quantities of 10 or more)
5.25" Mounting Kit, $3 additional)

4Mx9 80ns SIMMS, $199.95!!!

Break out those checkbooks!

Toshiba 3.5" Floppy Drive Problem

Well, last nite I finally found the fix for the ongoing saga of the Toshiba 3.5" floppy that insisted on being /D1 no matter what. On very close examination (glasses off, drive touching my nose) a surface-mount chip capacitor under the head-step motor showed printed '1' and '0' on opposite sides. Close probing with a jeweler's screwdriver revealed this "capacitor" to be a micro slide switch with an invisible handle -- trying to slide it towards the '0' side caused a satisfying click.

Also satisfied the drive, which is now quite happy as /D0 and can format hi-density clear out to all 80 tracks. If someone had just told me this 3 nites ago!

Full Address Decoding Explained (???)

Peripherals for the CoCo are all memory-mapped, meaning that they appear at some memory address (as opposed to Z80 or 8086, which has a separate set of "IO" addresses). There are basically two ways for a hardware device to recognize when it's address is being accessed:

1) "full address decoding" means that it looks at all 16 address lines on the expansion port to determine when it is being addressed.

2) There is an extra line on the expansion port called SCS (Spare Cartridge Select) which is only active (typically) when addresses $FF40-$FF5F are being accessed. By recognizing SCS, a device need only look at the bottom 5 lines to determine when it is being addressed.

The advantage of 1 over 2 is that when using a MultiPak, SCS is only active for one slot, which creates problems if you have multiple devices, since SCS must be "manually" switched from slot to slot.

A related issue concerns "ghosting", where a device does not look at all the necessary address lines, and so responds to more than one address. The original Tandy disk controller, for instance, only "needs" 4 addresses, but can actually be accessed at 32 different addresses. <sigh>

TOPS package...

I did some work on the TOPS package THE WHOLE THING is in one compressed tar file on garfield.catt.ncsu.edu (incoming directory). It is 7.6 MEG in size and uncompresses to 15 meg. (The sum total from smilodon was 7.9 meg and would not compress.)

Quote of The Month

Lawsuits, like assault rifles and nuke weapons, are great when you're going down and want to take a few of the bastards with you. If you want to go on living and working, they're a last resort.

How's this for "Vaporware"

The very minute I saw the first ad for Solaris at Infoworld, I grabbed the phone and asked them if they in fact did have a shipping product, as their ad plainly proclaims. (Something like, "Others talk about getting there, we're there now.")

"Sir that product is not available yet."

"Your ad says it is."

"I know that."

"Are you telling me that SUN SOFT is advertising a product with the words "here now", but it isn't available?"

"Yes"

"That's deceitful."

"Well, sir, really it isn't. The product will be available in July of 92."

"Your ad plainly says it is here now."

"I know that."

"Are you taking steps to pull the ad, since it is an obvious sham?"

"We don't see it that way sir."

"Exactly how would you describe an ad that contains an explicit lie?"

"Sir I'd prefer not to answer that. Would you like me to send you literature?"

"It seems clear that your ad campaign is trying to capitalize on the current confusion in the PC universe with respect to the fluzy tunes being sung by the various OS vendors. Is that in fact so?"

"Yes."

"So then why on earth would your first contact (the ad) with your new desired user base contain specific wilful

falsehood? How on earth do you expect to win customers if the first thing you say to them is a lie?"

"Sir the product will be available in about 11 months."

"And you call that "Here Now"?"

"Yes."

MM/1 Developers Wanted

All OS-9/OSK Programmers:

IMS is looking to expand the base of IDEA developers. IDEA stands for IMS Developers Association. IDEA offers several benefits for those that want to become registered developers with IMS, including the assistance of the most experience OS-9/OSK people in the country to aid you in your programming projects. In addition, IMS offers to buy up to $500 worth of your product once it is finished. That is, of course, if it is worth buying (grin).

IMS also offers free advertising to IDEA developers. If developers send a flyer to IMS, it will be included in all IMS mailings and in the boxes when MM/1s are shipped. Since the software product usually sells the hardware, IMS is very interested in getting high quality software to show off at demos and shows. Your potential for sales will be greatly increased.

What does one need to do to become an IDEA developer? Just send E-mail to me, or a letter to my address below. Include in your message what projects you have dreamed up, what you are currently working on, and what you would like to work on. You don't have to have an established reputation for excellent programming, nor do you even need to be an experienced hacker. All you need is a desire to develop applications for the future of OSK. "C" programmers, BASIC09 (Microware BASIC), PASCAL, Assembler.....any language will do.

We are looking to have a wide software base by the time the Chicago Fest rolls around in April so please be prompt in your reply. I am the IMS IDEA coordinator (slave driver is more like it) and will help in any way I can. Please remember that quite a few people have expressed an interest in porting UNIX utilities. These all already exist so if this is you intent, please try and think up something else. What we are looking for are new and innovative applications that make use of the MM/1s graphics and sound potential. Clones of existing IBM or MAC applications are OK, as long as a new "twist" can be put in to make it unique to the MM/1.

Thanks for your attention.

Mark Griffith

oooooooooo0000000000ooooooooo

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