USENET

Macintosh Programmer's Guide

Volume I

Thursday, October 18, 1990
(Updated Sun, 28 Dec 2014)

Compiled & Formatted by Matthew Xavier Mora

Table of Contents

Copyright Notice v

INTRODUCTION vii

Chapter 1 Miscellaneous Stuff 1

How to disable text edit from drawing (without flicker) 2
Algorithm for Random 2
Data structures for editing text 2
Data structures for editing text 3
Data structures for editing text 4
Better Docs/more proposals/useful sources of info 4
How do I mail to APDA 5
Apple deceives us + Useful code snippet [HitTestHook for TextEdit] 5
Simulating a Menu Bar with Pop-Up Menus (How To Abort out of Popupmenuselect) 7
The Eternal Question (Using TextEdit with HyperCard) 7
Good Think C books 8
Launching with Document List (with Source) 8
MacTCP problem (and a hack to fix it) 9
Mouse Handling (setmouse code in C) 10
Saving data from one launch to another 11
print variables on screen ? 12
print variables on screen ? 12
adding a document to the launch of an application? 13
Have you had problems with SetClikLoop? 14
need Apple-Double format help 15
Adding a document to the launch of an application? (desktop file notes) 18
Accessing Nubus board name 19
For GURU'S Only (Starting Point for serial comm) 21
Info about KeyCodes 21
Modifying the mouse input (SetMouse code in pascal) 21
(Novice) help with Undo function 22
What is the format of the scrapbook file 23
Programming the SCC (Code to poll the Macrecorder) 23
Width of popup menus 27
Script Manager Date Questions 28
Changing Radio Button Titles 28
Leading dashes in menu items. 29
Str255 --> 'STR ' how? 29
Where are disk icons kept? 30
How to tell when an appl quits? 30
How to get the current application (with code) 30
Random number Generator wanted. 31
Random number Generator wanted. 31
Random number Generator wanted. 32
Random number Generator wanted. 34
Random number Generator wanted. 34
Random number Generator wanted. 34
Random number Generator wanted. 35

Chapter 2 Code Resources (Inits,Cdevs,VBLs,...) 37

INITs and such... 38
INITs and such... 38
INITs and such... 39
How do you write a INIT in PASCAL? 39
Help! (jGNEFilter, GetNextEvent, events) 42
Info on Tail patches 42
Tail patches 43
Menu font list appearing as font (source to MDEF) 44
How can a CDEV "talk" to its INIT? 45
How do I install a driver with an INIT? (with Code) 46
cdev - INIT Data Exchange 50
cdev - INIT Data Exchange 51
CDEV/INIT Data Exchange, another way 51
cdev - INIT Data Exchange 52

CDEV/INIT Data Exchange, another way 53
Informative INITs (code incl. 53
24-Hour FKEY? 58
VBL tasks Think C 4.0 59

Chapter 3 Communications & Networking 61
Zones (how does one determine what zone one is in with code) 62
MacTCP programming with THINK C 63
Serial driver - Why is CTSHold being set? 63
Problem with Apple's distributed MacTCP dnr.c routine (bug has been fixed) 65

Chapter 4 Compilers & Development Environments 67
Ramblings about MacApp 68
To all users of ThinkC and MultiFinder... 69
MPW Pascal and self referencing 70
CClusters in Think C 71
LSP 3.0 bug (?) 71
global data in Think C 72
THINK Pascal interfaces (was SysEnvirons) 72
Alternative pkg interfaces for Pascal 73
Bug in THINK Pascal? My bug? 77
A possible bug in THINK C 4.0.1 77
32k arrays (in Think Pascal) 78
Multiple Inheritance for HandleObjects in C++ 79

Chapter 5 Dialogs & the Dialog Manager 81
Enter/Return in Modeless dialog 82
Informational dialog help (drawdialog stuff with code) 82
Password a'la AppleShare (Dialog Filter with code) 83
How can I draw contents of modeless dialog? 84
how can I draw contents of modeless dialog? 85
Floating Point #'s as Strings in Dialogs using Think C 3.0 85
Floating Point #'s as Strings in Dialogs using Think C 3.0 86
Query- Double lines on default button...How? 87
Query- Double lines on default button...How? 87
Password Dialogs--what is the best way to do them? 88

Chapter 6 File Manager 89
Finding amount of free disk space? 90
How to get pathname (with Code) 90
User items in SFdialog 91
SFGetFolder (with Code) 91
Repeat SetVol query 95
SFGetFolder 95
INIT getting it's file name... (with Code) 96
INIT getting it's file name...(with Code) 97
opendir() and readdir()(enumerate dir code in pascal) 97
Please help w/ PBSetCatInfo 99
FSRead hangs on the serial driver 99
FSRead hangs on the serial driver 100
The Prefered Way to Refer to Files... 100
SFReply.vRefNum -> PathName (pathname from vRefnum and Back in C) 101
Create a folder code (in Pascal) 103
How do I get a pathname from a working directory? (full Code) 105
How to get full pathname? 107
"SetVol-ing" to a folder in the System Folder... 107
Data Fork Use 108
Data Fork Of Currently Running APPL 108
Data Fork Use 108

Chapter 7 List Manager & LDEFS 111
List Manager advice 112
List Bummers Revisited (simple custom ldef with code) 112
List Bummers Revisited (and Debugging Help) 114
List definitions and fonts... 114
Cdevs and the List Manager 115

Chapter 8 Memory and the Memory Manager 117
ROM Unlocks handles (hlock and movehi slows programs down w/code) 118
Completion routine questions and anwsers 118
Segmentation (tips on using unloadseg) 119
Tips on unloadseg (and a story) 120
Handles and Virtual Memory (rewriting the Mem Mgr) 121
Setting up multiple heaps 121
NewPtrclear,NewhandleClear 122
NewPtrclear,NewhandleClear 123
StripAddress and pointer arithmetic 124
StripAddress and pointer arithmetic 124
llci Trap dispatch Table 125
Identifying real handles (with Code) 125
Identifying real handles 126
Need large array on Think C. 126
Need large array on Think C. 127
Need large array on Think C. (explanation on how to do it) 127
Need large array on Think C. 128
Identifying real handles 129

Chapter 9 Multi-Finder Madness 131
Context switching under MultiFinder 132
Inhibiting major switching under MultiFinder 132

Chapter 10 Printing & the Print Manager 133
Choose printer from software? 134
Generic Printing code (complete Source) 134

Chapter 11 QuickDraw™ & Graphics 139
Copybits and the colortable 140
Piccomment and technote 175 140
Sync Drawing (with Code) 141
THINK C & 32-bit quickdraw (interfaces for 32bit quickdraw) 143
THINK C & 32-bit quickdraw (.c) (interfaces for 32bit quickdraw) 145
How do I get the slot of the main screen? 149
Drawing to an off screen bitmap (with code) 150
Color Quickdraw and copybits glitches (ps to colorqQD with Code) 151
Finding the size of the screen under Multifinder/Color QD (gdevices) 156
Bitmap not showing up in scrapbook (cliprect) 157
Finding the size of the screen under Multifinder/Color QD 157
Finding the size of the screen under Multifinder/Color QD (with C code) 158
Marquee Help? (with Code) 159
How to get the QD globals outside of an application 162
Marquee Help? 163
Finding the size of the screen under Multifinder/Color QD 163
Finding the size of the screen under Multifinder/Color QD 163
Finding the size of the screen under Multifinder/Color QD 164
PICT resource to BitMap (with code) 165
Writing large PICT files??? 166
How's the screen cleaned after _SysError? 166
Optimizing Copybits (good Information with code) 167
Fast Copybits suggestions needed, other questions 169
How does one set up an offscreen buffer 169
Rotate Bit Map CCW (full source code) 171
Aligning bitmaps? (with code) 182
Aligning bitmaps? 183
Aligning bitmaps? 184
ThePort and TENew (problem with font) 184
FatBits,thinbits and MapPt() 185
32-bit QuickDraw scaling behavior 185
Need help creating a Quickdraw picture 186
Still need help creating a QuickDraw picture 187
Need help animating with 24 bit card 187
Setting up CLUTs - Advise??????? 188
Bad pixmap data written by OpenPicture & CopyBits 188
Fading ...(dissolve effect) 189
Drawing all over the desktop 189

PICT hacking question ... 190
Wanted- a window that comes up in black ... 190
Wanted- a window that comes up in black ... 191
Dragging Bitmaps? ... 191
Wanted- a window that comes up in black ... 192
Dragging Bitmaps? ... 192
2-16-256-million color mode ... 192
Reading PICT's revisited. ... 193
How to read a Pict into a scrollable window ... 197

Chapter 12 Resources & the Resource Manager ... 199
Creating a Resource Fork ... 200
Interesting problem whe GetResource('dctb',id) [***LONG***] ... 200
Puzzling Resource Problem ... 201
Interesting problem with GetResource('dctb', id) ... 201
Interesting problem with GetResource('dctb', id) ... 202
Altering the Resouce fork ... 202
OpenResfile Question ... 203
How to Openresfile [Summary] ... 203
OpenResfile Question ... 205

Chapter 13 Sound & the Sound Manager ... 207
Need help with the new sound manager! (with code) ... 208
Synchronous sounds using the Sound Manager ... 209
Synchronous sounds using the Sound Manager ... 209
How to get rid of clicking? ... 210
Audio Phone Dialer Software ... 211
Simultaneously played digitized sounds ... 212

Chapter 14 Windows & Grafports ... 213
Checking the validity of a refCon (window refcon) ... 214
Floating Windows ... 215
How do you find a window's location? ... 215
How do you find a window's location? ... 216
How do you find a window's location? ... 216

Chapter 15 Articles & Notes ... 219
Macintosh One-Liners ... 221
Scheme to Manage a "Windows" menu ... 236
How to write an INIT in Pascal ... 241
How do you play Asynchronous Sound? ... 243
Default 2.1 CDEF ... 246
ToolBox Gotchas ... 255
How do you Hide the menu bar? ... 257
BitMap Rotation in C (and support routines) ... 260
INIT Skeleton Code ... 269
New Volume Scanning Algorithm ... 275

APPENDIX A ASCII Table ... 277

APPENDIX B Error Codes ... 279

Authors and Credits ... 281

Copyright Notice

Exception Clause

Any articles, questions, answers or any other miscellaneous text written by Tim Maroney, are to be considered in the "public domain" and you may use them in any way you see fit. You DO NOT have to give him any credit or any mention in your programs or documentation, for any of the information he unknowingly provided in this guide. He specifically stated that he wants to keep his posting in the public domain so that they will help the most people.

To get a copy of this volume in printed form send $19.00 to:

MXM Designs
Attn: UMPG Printed Version
39075 Carmel ct.
Fremont, CA 94538

This volume is also available on a disk in Microsoft Word 4.0 format.

To get a copy of the disk version send $5.00 to:

MXM Designs
Attn: UMPG Disk Version
39075 Carmel ct.
Fremont, CA 94538

PLEASE MAKE CHECKS PAYABLE TO:
MXM Designs

INTRODUCTION

Welcome to USENET Macintosh Programmers Guide Volume I

This is the first attempt at creating a document for the readers of USENET that deals with Macintosh programming. The first section of this volume is a collection of USENET postings that have appeared in the famous "comp.sys.mac.programmer" newsgroup. This section includes articles on things that you should know when programming the Macintosh, common questions and answers that might help you when you have a problem or general knowledge you should have to prevent you from making a mistake. I started this as a guide for myself because so much information gets posted to the net, its hard to remember it all. Other people seemed interested in the guide so I decided to publish it.

The second part of this guide are articles that people have donated either to me directly when they heard that I was going to publish my guide or articles that have appeared on the net. Hopefully when I get around to making version two of this thing, more people will donate sample code to make this guide more complete. I hope to either include a beginner's section or maybe make it a totally separate volume.

So enjoy, and I hope that this guide helps you as much as it has helped me.

Any questions, comments, or code segements can be sent to me at the addresses listed below. I will include them in the next update.

Matthew Xavier Mora
Internet mxmora@unix.sri.com
Applelink D5438
QuickMail Matt_Mora@qm.sri.com
US Mail 39075 Carmel ct. Fremont, CA 94538

I would like to thank all those people that sent me articles and have donated their free time to answer questions on the net. It is you who are doing a great service to the Mac programming community. I am only the messenger.

Chapter 1 Miscellaneous Stuff

How to disable text edit from drawing (without flicker) 2
Algorithm for Random 2
Data structures for editing text 2
Data structures for editing text 3
Data structures for editing text 4
Better Docs/more proposals/useful sources of info 4
How do I mail to APDA 5
Apple deceives us + Useful code snippet
[HitTestHook for TextEdit] 5
Simulating a Menu Bar with Pop-Up Menus
(How To Abort out of Popupmenuselect) 7
The Eternal Question (Using TextEdit with HyperCard) 7
Good Think C books 8
Launching with Document List (with Source) 8
MacTCP problem (and a hack to fix it) 9
Mouse Handling (setmouse code in C) 10
Saving data from one launch to another 11
print variables on screen ? 12
print variables on screen ? 12
adding a document to the launch of an application? 13
Have you had problems with SetClikLoop? 14
need Apple-Double format help 15
Adding a document to the launch of an application?
(desktop file notes) 18
Accessing Nubus board name 19
For GURU'S Only (Starting Point for serial comm) 21
Info about KeyCodes 21
Modifying the mouse input (SetMouse code in pascal) 21
(Novice) help with Undo function 22
What is the format of the scrapbook file 23
Programming the SCC (Code to poll the Macrecorder) 23
Width of popup menus 27
Script Manager Date Questions 28
Changing Radio Button Titles 28
Leading dashes in menu items. 29
Str255 --> 'STR ' how? 29
Where are disk icons kept? 30
How to tell when an appl quits? 30
How to get the current application (with code) 30
Random Number Generator wanted. 31,35

From: unknown
Subject: Re: How to disable text edit from drawing (without flicker)

To modify a textedit record without annoying selection rectangle flicker, call:

```
TEDeactivate()    /* turns off the text edit cursor */
TESetSelect()  /* the part you want to erase */
TEDelete()     /* erase it */
TEActivate()   /* turn the cursor back on */
```

●●●

From: john@trigraph.uucp (John Chew)
Subject: Re: Algorithm for Random

In <8505@spool.cs.wisc.edu> engber@thylacine.CS.WISC.EDU (Mike Engber) writes:

> >I'm looking for the details of how Random is implemented in QuickDraw
> >so students of mine that are not working on Macs can generate the same
> >sequence of random numbers to use in testing their programs.

Random() generates random numbers in the range [-32767,32767] using a 32-bit seed stored in the QuickDraw global randSeed.

Each time it is called, it does the following:

1. Multiply randSeed by 0x41A7 to form a 47-bit product.
2. Take bits 31 through 46 (MSB) of the product and add them as an unsigned value to bits 0 (LSB) through 30 of the product to form the new seed.
3. Return the low sixteen bits of the product as the random number, except when they are the value 0x8000, in which case return zero.

Here is some Lightspeed C code to clarify the above:

```
-----
#define A ((unsigned long)0x41A7)
#define High(x) ((unsigned int)HiWord((x)))
#define Low(x) ((unsigned int)LoWord((x)))
int myRandom(void);

int myRandom()
	{
	unsigned long temp;

	/* wish i had 64-bit data... */
	temp = A*High(randSeed) + High(A*Low(randSeed));
	randSeed = ((temp & 0x7fff) << 16) + High(temp<<1) + Low(A*randSeed);

	if (Low(randSeed) == 0x8000)
		return 0;
	else
		return Low(randSeed);
	}
```

●●●

From: pete@titan.rice.edu (Pete Keleher)
Subject: Re: Data structures for editing text

> What sort of data structures are generally used for representing text in an
> editor or word-processor? Obviously TextEdit is an alternative, but it's slow
> (as far as I've heard) and imposes size restrictions.

Not only does TextEdit impose size restrictions, but it's updating leaves a lot to be desired. One of my goals was to get the editor to look and feel as solid as Think C's editor, but not be bare-bones.

Good question. I have looked at source for two editors and have written one of my own. The best data structure that I have heard of would be "chunks". A chunk would be a chunk of text, maybe 256 bytes, typically half full. The chunks are then linked together in a linked list. Insertion/deletion usually involves moving only some of the bytes in the current chunk. Sometimes, of course, you must split chunks or combine them. There is usually no array of line pointers, except maybe for the lines currently being displayed. Two important data structures could be declared something like:

```
typedef struct Chunk {
    struct Chunk   *next, *prev;
    short      len;
    char       text[256];
} Chunk;

typedef struct {
    Chunk   *chunk;
    char    *cp;
} mark;
```

In my editor, I use one large chunk for the file image that is read from disk,along with an array of line pointers into the file image. When a line is modified, I allocate a chunk and go from there. When lines are inserted/deleted I need to move all of the line pointers after the line in question, but in practice this is cheap. The overhead of having data in the file image in memory but no longer used is also cheap. This is definitely NOT the way to do it if you're designing from scratch, but I made some poor decisions in the beginning (basically I put off the decision, and by the time I could see where to go, I couldn't do it without re-writing everything). The up side is that:

1) In the case where relatively little is modified, (say less than 30%), memory costs are low.
2) It is very fast. In particular, I can access any line as fast as any other. as opposed to the straight chunk approach where you have to search through all of the lines between the start of the file and your current position just to find out what line you are on.

--
Pete Keleher pete@titan.rice.edu

●●●

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re:Data structures for editing text

The linked list suggestions made so far may be fast enough for a word processor, but the Memory Manager will really bog down when you have a lot of these chunks allocated. If you want better performance, it would probably be better to pre-allocate a large buffer and treat nulls as non-characters, the "holy buffer" approach. This will completely avoid Memory Manager overhead, but may require you to move more data explicitly. You can optimize it in various ways, as any second-semester data structures textbook will tell you. You can leave extra space to make insertions faster, or use only as much space as the user types if you feel deletions are more important.

The subject is not trivial and I recommend some research before you begin. I just plunged ahead with a relocatable linked list approach for one of my terminal emulators and I wound up with something that couldn't keep up with 2400 baud on an 8MHz 68000. On the other hand, it was a lot faster for data input than the MPW Shell. So I guess my main advice is -- be prepared to throw away your first stab at it. Make sure knowledge of your data structures is confined to only those files that need it, so that you can change them when you find that your first algorithm bogs down in certain situations. Be sure to test speed with respect to operations like large pastes and very fast typing. And familiarize yourself beforehand with the various buffering strategies that can be found in the textbooks. Most importantly, be sure to leave yourself a way to buffer large files on disk.
--

Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: pepke@loligo (Eric Pepke)
Subject: Re:Data structures for editing text

It just so happens that I am currently working on a text editor for programming right now. I am using a single buffer with a single gap using long integers to remember the offsets to the gap, which I think is similar to the way Capps used to do it. (If Capps were still a supported product, I would probably be using it.)

A reasonably large gap is kept at the current insertion point. Normal character insertions just go into the gap and make the gap one character smaller. Deletions of single characters just increase the size of the gap by one. The only times that text has to be moved are when a new gap is allocated or the old gap is removed. The idle routine tries to do this at unobtrusive times.

Scroll bar handling is done as an integral part of the text editor. I strongly recommend that you do this rather than follow TextEdit's method. Scroll bar handling for long integers is very easy to do cleanly if you have enough information available. If you try to tack it on outside of the editor a la TextEdit, it is very difficult to do cleanly.

Eric Pepke INTERNET: pepke@gw.scri.fsu.edu

•••

From: ari@eleazar.dartmouth.edu (Ari Halberstadt)
Subject: Re: Better Docs/more proposals/useful sources of info

In article <1989Nov1.232424.8861@agate.berkeley.edu> silverio@brahms.berkeley.edu (C J Silverio) writes:

> >I am sick to death of trying to code Mac applications with the
> >existing documentation.
> >
> >Consider, for example, the case of TextEdit. It was originally
> >documented in IM-I, then there were a few tech notes on it, then it
> >was revised in IM-IV and again in IM-V, with a spate of additional
> >tech notes following. One needs to obtain, read, and piece together
> >all of these bits of info to use TextEdit with maximum effect. Why
> >hasn't anybody done this piecing together already?

Yes yes...the project I've been working on took me 1/4 times longer cause of lousy docs. At least Apple cleaned up their mess in AU/X...they have about 20 manuals, ring bound, small, and the binder rings are ROUND!!! Yes, you don't have to push 600 pages to get to the index!!! (or maybe I'm fantasizing, maybe in fact the binder rings are rectangular...). Each manual actually deals with what it says: there's one about text editing, one about games (ya, they put rogue on the mac :-), etc.

The best things to date:

1. Inside Mac DA. The version I have goes up to vol IV, which is most of what you need. Saves endless trips to the bookshelf. It's amazing how 4 volumes can be shrunk to 6x5 inches.

2. OnBase. When you need a prototype, or the name of a field in some obscure structure, OnBase is the one!

3. HyperCard technotes stack. Just the regular tech's, but in stack form.

4. MAC DTS sample code. Virtually useless to me, since I've had to learn all that junk myself...oh well. The examples just aren't terribly serious. And even on Phil and Dave's excellent CD I could find no sample DA code, so again I had to plow through three very poorly documented sections of IM. Turned out, writing a DA is simpler than an application, once you know how.

That's about it. Oh, someone asked about handles, but I couldn't get through to him. If you're still out there:

```
**hndl = SomethingThatFoolsWithMemory();
```

Don't do that: THINK C evaluates the pointer first, then calls the function, so if memory moves, you're in trouble. Similarly, don't do

```
SomethingThatFoolsWithMemory(&(**struct_hndl).element);
```

For the same reason.

-- Ari Halberstadt

•••

From: rickf@Apple.COM (Rick Fleischman)
Subject: Re: How do I mail to APDA

In article <32370@ucbvax.BERKELEY.EDU> thom@dewey.soe.berkeley.edu.UUCP (Thom Gillespie) writes:

> >What kind of mailer do I use to send email to APDA. I've tried
> >user_name@apple.com
> >
> >and I get returned mail?
> >
> >Any ideas? Thanks.
> >
> >--Thom Gillespie

You can send e-mail to APDA through AppleLink, Apple's internal e-mail system. You can do this by sending mail to: APDA@applelink.apple.com This is a gateway to AppleLink and APDA will receive your messages.

If you have any trouble, send an e-mail to me at rickf@apple.com and I will forward your message on.

Have fun!

Rick Fleischman Developer Channels/APDA Apple Computer, Inc.

•••

From: ari@eleazar.dartmouth.edu (Ari Halberstadt)
Subject: Apple deceives us + Useful code snippet [HitTestHook for TextEdit]

Hello all ye mac hackers. I recently spent countless hours figuring out what Apple didn't tell us. I've been working on adapting TextEdit for a specific application, and needed to write most of the hooks for the new styled text edit in system 6.0. Following is a quote from TechNote #207, describing the HitTestHook:

```
==============================
TEHitTestHook

This routine is called to determine the character position in a line given the horizontal offset, in pixels, from the
beginning of a line. The default action is to call Pixel2Char and return. For more information, see the description of
Pixel2Char in the Script Manager chapter of Inside Macintosh Volume 5.

On entry: D0  length of text to hit test       (word)
          D1  pixel offset from start of text  (word)
          A0  pointer to start of text         (long)
          A3  pointer to the TextEdit record   (long)
          A4  handle to the TextEdit record    (long)

On exit:  D0  pixel width to last offset       (low word)
              Boolean = TRUE if a character    (high word)
              offset corresponding to the
              pixel width was found.
          D1  character offset                 (word)
          D2  Boolean = TRUE if the pixel      (word)
              offset falls within the left side
              of the character.
==============================
```

Notice the words "the default action is to call Pixel2Char and return". If Apple actually succeeds in doing this, I'll be truly amazed. If you look at the code I ended up writing, you'll see there are many special cases with which I had to

cope. Why Apple purposefully mislead us, I have no idea. As to why Apple could not simply show us some sample code, I haven't a clue.

==========

Following is the hit test hook which I finally ended up writing. It was written with THINK C 4.0, and will run correctly even from an XCMD or device driver. To use it, you must install it using TECustomHook, which is described in TechNote #207. The code should run ok in MPW, if you remove the calls SetUpA4() and RestoreA4() and adapt the assembly code snippets. If you use this code from a code resource in THINK C, you must do a RememberA4() somewhere in the same file as this function is in, and at a time when the value of A4 is correct. Currently, the function does the same things that the default HitTestHook does, but, of course, you may enhance this function as needed.

Disclaimer: I'm not sure the code is 100% correct, but it seems to live very happily with good old TextEdit. Notice that you can't run profiling when using this hook, since the THINK C profiler inserts a function call at the start of each function in your program, which means the registers are destroyed when it returns.

==========

```
/* Please acknowledge source if you use this code in your programs... */
void
HitTestHook()
{
	int	length;		/* length of text to hit test */
	int	poffset;/* pixel offset from start of line */
	Ptr	text;  /* pointer to start of text */
	Style  hsStyles;/* the hot spot styles */
	Boolean		leftSide;/* true if pixel width falls within left
				side of a character */
	int	offset;		/* offset of character that pixel is closest to */
	GrafPtr		currentPort;/* the current port */
	int	width;		/* width of text */

	/* get stuff from registers, and save all other registers */
	asm {
		movem.l		d3-d7/a0-a3, -(a7)
		move.w d0, length
		move.w d1, poffset
		move.l a0, text
	}
	SetUpA4();

	/* do interesting stuff, such as disabling certain text styles */

	/* calculate offset into text corresponding to pixel */
	offset = Pixel2Char(text, length, 0, poffset, &leftSide);
	/* offset must [usually] be incremented; this copes with clicks in most
	areas of the text */
	if (leftSide && offset+1 <= length)
		offset++;
	/* this copes with click after last character in file */
	if (offset > 0 && offset == length && text[offset] != '\r')
		leftSide = false;
	/* this copes with click before first character of a line */
	if (offset == 0 && text[-1] == '\r') {
		leftSide = true;
		offset++;
	}
	width = Char2Pixel(text, offset, 0, poffset, 1);

#ifdef DEBUG
	/* if you enable this code, you can see some pretty
	interesting things! If you don't have THINK C, this may
	not work very nicely (THINK C lets you do printf's in an
	inactive window).
	*/
	printf("width=%d, poffset=%d, leftSide=%d, offset=%d\n",
```

```
                width, poffset, leftSide, offset);
        { char str[256];
                strncpy(str, text, length);
                str[length] = 0;
                if (str[length-1] == '\r')
                        str[length-1] = '\n';
                printf("text='%s', length=%d\n", str, length);
        }
        SetPort(currentPort);
#endif

        /* save return values */
        if (width < poffset)
                asm { move.w #false, d0 }/* no offset was found */
        else
                asm { move.w #true, d0 }/* else, set d0 to true */

        asm {
                swap    d0              ; stuff condition into high word of d0
                move.w  width, d0       ; set low word of d0 to pixel width
                                        ; to the last offset
                move.w  offset, d1      ; store character offset in d1
                move.w  leftSide, d2    ; set d2 to true if pixel falls on
                                        ; left side of a character
        }

        /* cleanup and exit */
        RestoreA4();
        asm {
                movem.l         (a7)+, d3-d7/a0-a3
        }
}
--
```

Ari Halberstadt

•••

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: Simulating a Menu Bar with Pop-Up Menus (How To Abort out of Popupmenuselect)

I got an answer via e-mail very quickly to my posting. Unfortunately, I didn't save the writer's name or address, but it was one of the authors of a program called Nisus. In case anyone else wants to do this, the answer is: MenuSelect or PopUpMenuSelect can be aborted simply by posting a mouseUp event from MenuHook. It will terminate as soon as it sees a mouseUp in the queue; it will not wait for the Button routine to return false. Of course, this makes perfect sense, since the mouse could have been clicked again already. So there is no need for a trap patch on Button.

Thanks to my correspondent for providing this information.
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

•••

From: minow@mountn.dec.com (Martin Minow)
Subject: Re: The Eternal Question (Using TextEdit with HyperCard)

In article <11887@phoenix.Princeton.EDU> bskendig@phoenix.Princeton.EDU (Brian Kendig) writes:

> >Good points; let me just clarify a few things I said:
> >
> >Is there any code out there to show me how to 'simulate' TextEdit,
> >then, for a large amount of text? This doesn't sound like child's
> >play...

It's not. The source for my TextEdit clone (to appear this spring in MacTutor) fills most of a floppy. And it doesn't change the underlying -- inefficient -- organization of the data (this is left as an exercise for the student :-).

What I'd do is to write a separate program that breaks the text into paragraph-sized chunks (each paragraph <= 32K bytes) and store each paragraph in a database of sorts. I suppose you could use a resource fork, though there are more efficient (and less general) organizations.

Then you need to build a database of paragraphs with the database designator (location in the file, text size, and formatting information). Your "build" program would do this by reading each paragraph into a TextEdit record and calling TEGetHeight (IM-V) to get the formatting information. Now, to display the text, you read enough paragraphs to fill the screen and write them in a window (Don't use TextEdit for this.) Note, by the way, that you'll have to handle styles and multiple fonts yourself. This isn't that difficult to do.

As a previous reply mentioned, you will have to "jacket" the scroll bars so the range you present to the control manager is within the short integer range permitted. Keeping the "true" range in floating-point would probably be sufficient. I.e., if the range is 0-1.0, you would do something like

```
SetCtlMin(..., 0);
SetCtlMax(..., 32767);
SetCtlValue(..., (short int) (true_value * 32767.0));
```

Your scroll/display module would read paragraphs from the database as needed. You probably want to manage the paragraph storage yourself, rather than relying on the Macintosh storage routines.

You also mention searching. If you have a large chunk of text, consider building a search-hash array for each line (or one per paragraph and one per line). Such an array would have, say, 256 bits per line. Each word in the text would hash to a single bit, which would be set in the hash array. Your search word would then be hashed. To search the database, you need only look at those paragraphs/lines which have the search bit set. There is a lot of room for experimentation here: what is the right number of bits, should you set more than one bit, and require the pre-search to match both, etc. There might be something on this in Knuth V3, or other books on searching. I remember reading about this technique in a CACM article in the late 1960's or early 1970's.

> >The text involved here is indeed read-only; I had overlooked that!
> >This makes life a *little*, if not greatly, easier.

This is the key to a fast system: do the work up front in a "database compiler" so the user is not delayed.

Good luck.

Martin Minow

•••

From: jmann@bigbootay (Jim Mann)
Subject: Re: Good Think C books

Try _Macintosh Programming Primer_ (subtitled "Inside the Toolbox Using THINK's Lightspeed C") by Dave Mark adn Cartwright Reed.

I t provides a good introduction to programming on the Mac and using the toolbox routines. It also has information on using Lightspeed C and on using some of the standard Mac development tools (such as ResEdit).

•••

From: ba0k+@andrew.cmu.edu (Brian Patrick Arnold)
Subject:Re: Launching with Document List (with Source)

Hello,

Aaron Wohl once helped me with this problem. I get the feeling Apple may change its mind on how this works when System 7.0 arrives. You need tech notes #52 and #126 for details on (sub)launching. Right before sublaunching, you need to poke a handle into low memory:

```
{Portions (c) 1986-1988 by Apple Computer, Inc. All rights reserved.}

CONST
  { Location of Low Mem Global: handle to hold app parameters }
  { sort-of-documented in ToolEqu.asm MPW Assembly equate file }
 AppParmHandle =  $AEC;

TYPE
  { semi-documented finder info in launching an app with a file - MAY
    BREAK on future Systems - but explained in IM #2 Segment Loader }
 HApParms = ^PApParms;
 PApParms = ^RApParms;
 RApParms = RECORD
  printFlag :  INTEGER;   { use appOpen or appPrint constants
                          from Segment Loader }
  numFiles  :  INTEGER;
  appFiles  :  ARRAY[1..1] OF AppFile;
 END;
 LongPtr = ^Handle;       { hack to poke low memory global location }

PROCEDURE SubLaunch;
  { launch an app with files as if from Finder }
VAR
 MyAppParms   : HApParms;
 AppParmLoc   : LongPtr;
 fooVRefNum   : INTEGER;
BEGIN
   { don't ask - use SFGetFile or another means }
  fooVRefNum := MagicIncantationForGettingFileVRefNums( 'Foo' );

   { Set appParms to launch and open files }
  myAppParms := HApParms( NewHandle( SIZEOF( RApParms ) ) );
  WITH myAppParms^^ DO BEGIN
   printFlag := appOpen;
   numFiles  := 1;

   appFiles[1].fName   := 'Foo';        { my hack }
   appFiles[1].vRefNum := fooVRefNum; { you worry about this }
   appFiles[1].fType   := 'TEXT';                   { and this }
   appFiles[1].versNum := 0;
  END;

   { poke the low mem }
  AppParmLoc := LongPtr( AppParmHandle );
  AppParmLoc^ := Handle( myAppParms );

   { Do the launch as per TechNotes }
END;
-----------------------------------
```

People at Apple are encouraged to give UITP warnings as needed.

- Brian

•••

From: dorner@pequod.cso.uiuc.edu (Steve Dorner)
Subject: MacTCP problem (and a hack to fix it)

We've found a problem with MacTCP. Specifically, it advertises a bad TCP Maximum Segment Size (MSS) to some hosts that are not on the same class B subnet as our Gatorbox.

THE FACTS

Our setup is as follows:

Gatorbox is on a class B subnet, 128.174.33. Macintoshes are connected to the gatorbox by PhoneNet. Pequod (a NeXT machine, but that doesn't matter) is also on 128.174.33. Uxc (a 4.3bsd-tahoe VAX, but that doesn't matter, either) is on 128.174.5, a subnet a gateway or two away.

And this is what we found (by snatching packets off the ethernet):

To pequod, MacTCP advertised an MSS of 546. Add 40 bytes for TCP/IP headers, and you wind up with a 586 byte IP datagram, which is the maximum possible ip datagram that can be forwarded on the PhoneNet. And, in fact, pequod sent maximal packets of 586 bytes on its connections with MacTCP. So far, so good.

To uxc, however, MacTCP advertised an MSS of 576. Add 40 bytes for TCP/IP headers, and you get a 616 byte IP datagram, which is TOO big. Since gatorboxes don't fragment oversize IP datagrams, a packet that size will never be received by MacTCP, and the connection will eventually die. And this is exactly what we saw. Connections worked fine until there was a lot of data to be sent to the Mac, at which point uxc sent a 616 byte datagram, and the connection died.

While we did not do packet analysis on all the connections, we saw hung connections to every single host we tried (11 different hosts, from many different vendors) that was on a subnet of 128.174, but NOT on 128.174.33. The only host on 128.174.33 worked fine, as did both the hosts we tried that were on totally different networks (i.e., not 128.174).

THE SPECULATION

MacTCP advertises too large an MSS to hosts which are on the same class B network, but not on the same class B subnet. It advertises proper values to hosts either on the same class B subnet, or on entirely different networks.

THE HACK

Fortunately, a little disassembly and trial and error led to a patch that makes the problem go away. Find the hex byte string "337c02040014" (at an offset of around 0x6500, give or take 0x200) in the MacTCP document, and change it to "337c01010014".

This has caused MacTCP to function adequately for our setup for all the hosts I tried. I'm not POSITIVE what this patch does (I suspect it just nukes the MSS advertisement altogether, but I haven't verified that). It resolves our problem, and I'll look no further.

THE REAL SOLUTION

1. Apple needs to fix MacTCP to advertise the MSS correctly to different subnets. This is the OPTIMAL fix, since IP fragmentation is unpleasant at best.

2. Cayman needs to fix the GatorSoftware to fragment large IP datagrams. (Michael Haag [from Cayman Technical Support] assures me Cayman has done so in the next release, due out 1Q90.)

--
Steve Dorner, U of Illinois Computing Services Office

●●●

From: cbm@well.UUCP (Chris Muir)
Subject: Re: Mouse Handling (setmouse code in C)
Summary: Here's SetMouse again.

In message <1989Dec7.183658.18691@Solbourne.COM> tonyj@marvin.Solbourne.COM (Tony Jackson) writes:

> >I am trying to attach a device to the serial port of a Mac which could
> >be used as a replacement for the mouse.
> >
> >Talking to the device is simple. The problem I'm having is finding any
> >information as to the location of the globals (?) or trap to use to
> >update the mouse coordinates. Shoving a pseudo-mouse event into the
> >queue doesn't work as that does not update the pointer location on the screen.

This seems to come up every few months. I've posted bad Pascal code for this SetMouse routine in the past. Here's a Think C version:

```
/* ------------------------ SetMouse ------------------------- */
/* some dangerous low-memory-global equates */
extern  Point   MTemp          :      0x828;
extern  Point   RawMouse       :      0x82c;
extern  Point   Mouse          :      0x830;
extern  Byte    MBState                :      0x172;
extern  int     CrsrNewCouple  :      0x8ce;  /* both New & Couple */
extern  Byte    CrsrNew                :      0x8ce;
extern  Byte    CrsrCouple     :      0x8cf;

#define         Couple         0xff;  /* value for CrsrCouple */
#define         Uncouple       0x00;  /* value for CrsrCouple */

void    SetMouse(where)
Point   where;
{
long    finalTicks;
        LocalToGlobal(&where);   /* Get ready to store mouse position */
        RawMouse = where;        /* into RawMouse */
        MTemp = where;           /* and MTemp */
        CrsrNewCouple = 0xffff;  /* Hit CrsrNew & CrsrCouple */
        Delay(5, &finalTicks);   /* let the cursor catch up */
}       /* SetMouse */
```

Chris Muir

●●●

From: 6600pete@hub.UUCP
Subject: Re: Saving data from one launch to another

From article <975@manta.NOSC.MIL>, by lulue@manta.NOSC.MIL (Dan Lulue):

> I need to save the location of a data file from one launch to another.
> The user points out the file's location via Std File, and my program
> "remembers" it for the next time. I am constrained to using UNIX style
> string pathnames. Is it preferable to save the string in a 'STR ' resource in
> application's resource fork, or in the data fork?

Neither. If you ever want your application to run on a network server, you can't do this, because the resource manager doesn't know about multiple simultaneous users. The file manager (data fork strategy) knows, but you don't want to use it, because the user wouldn't be able to count on settings lasting if the app lived on a server. (Plus, you'd have to do record locking for one little config string, and it would be a pain.)

A better way to do this is to save a settings file in the System Folder. You can find out where that is with SysEnvirons.

> I tried the resource fork by creating a 'STR ' resource, doing a GetString
> (which does a GetResource('STR ',ID), changing the string pointed to by the
> handle, calling ChangedResource , then WriteResource, etc. However,
> the new string never shows up in the resource fork (ResEdit view) after
> execution (the dummy string I placed there originally is still there).

GetResource, LoadResource (depending on various settings), HNoPurge,make your change, ChangedResource, WriteResource, HPurge. This sequence is covered in IM I under ChangedResource.

> Chernicoff also suggests that the data fork is a good place to store
> this sort of data.

Chernicoff's getting old. Do you have the second edition? Or am I thinking of another series of books? Somebody?

> how does one get the vRefNum of the directory the application resides in?

Do a GetVol on startup. You'll get a working directory refNum.
-
Pete Gontier 6600pete@ucsbuxa.ucsb.edu

•••

From: dowdy@apple.com (Tom Dowdy)
Subject: Re: print variables on screen ?

In article <1837@ultb.isc.rit.edu> sfm3166@ultb.isc.rit.edu (S.F. Modi) writes:

> say printf("xxx") as it defaults to the compiler window. So how do you
>print a variable ? for example I want to print

Well, this example might be a bit off base from what you want, but it certainly can be changed into what you want. I've seen versions of this called wprintf() that output into a scrolling window. I personally usually prefer to see things in Macsbug, because usually this code is only in for debugging.

Follows is code to output printf() style into Macsbug. I use this with MPW and <stdarg.h>. Requires linking with C libraries. The runtime routines require A5, thus not too useful in VBL tasks and so forth (sorry, Tim). Uses 256 bytes of stack space for the temp string. Haven't tried this sort of thing with Lightspeed, maybe someone can convert it and let
the world know.

Hope this is useful to some folks out there...

```
/* ------------------------------------------------------ */
void debugprintf(char *sFormat, ...)
/*
   Stolen from Geoff Coco.
   Take a printf style list of args, pass them on and debugstr them.
   Call as you would printf.
*/
{
Str255   sOut;
va_list  pArgs;

va_start(pArgs, sFormat);
vsprintf(sOut, sFormat, pArgs);
c2pstr(sOut);

DebugStr(sOut);

} // debugprintf
```

Tom Dowdy Internet: dowdy@apple.COM

•••

From: beard@ux1.lbl.gov (Patrick C Beard)
Subject: Re: print variables on screen ?

In article <33467@ucbvax.BERKELEY.EDU> oster@dewey.soe.berkeley.edu.UUCP (David Phillip Oster) writes:

>In article <5960@internal.Apple.COM> dowdy@apple.com (Tom Dowdy) writes:
>>I personally
>>usually prefer to see things in Macsbug, because usually this code is only
>>in for debugging.
>
>Tom is right, Here is dprintf() for THINK C.:

David's version assumed some things about vsprintf. Here is the more conventional way, and this does work under THINK C V4.0:

```
#include <stdarg.h>
void dprintf(char *sFormat, ...)
{
       char          sOut[256];
       va_list  pArgs;

       va_start(pArgs, sFormat);
       vsprintf(sOut, sFormat, pArgs);
       CtoPstr(sOut);
       DebugStr(sOut);
}
```

- Patrick Beard, Macintosh Programmer (beard@lbl.gov) -

•••

From: doner@henri.ucsb.edu (John Doner)
Subject: Re: adding a document to the launch of an application?

In article <1085@crash.cts.com> alen@crash.cts.com (Alen Shapiro) writes:

> >I think there was a thread a little while ago about how one should
> >add a document list to appparmhandle global area. I missed it!!
> >
> >Could someone mail me a summary please. I can launch an application

I missed that thread too, but here's the source and documentation for a program I wrote a couple of years ago that does what you want, or something like it.

This little program launches an application with a specified document. This is done by setting up the Finder Information, the data structure pointed to by the low-memory global AppParmHandle. Read about it in the Segment Loader chapter of IM. Apple provides routines for accessing the Finder Information, but none for creating it, since they thought that would always be done by the Finder. The Finder Information consists of two words giving a message and a count of the number of AppFile records following. Each record contains info on a file, including vRefNum, file type, and file name.

This program has a STR# resource, ID 100, with three strings: the full pathname of the application to be launched, the full pathname of a file to be listed in the Finder Information, and the File Type of that file. It creates the Finder Information and launches the application. It oughtn't be hard to modify this so that multiple files could be listed in the Finder Information. And, you could just as well construct pathnames and get file types in some other way than reading them in from a resource.

Note: The AppFile data structure is a record of which the last field is the filename, a Pascal string. These are declared as arrays of 256 bytes, and that's what you'll get if you create them in the normal way in Pascal or C. However, the assembly-coded Finder won't waste all that memory on short strings; instead each record begins right at the actual end of the string in the previous record. So you can't regard the Finder Information as an array of equal-sized records; you have to use the string length in one record to find the beginning of the next one. That's why one should use the Apple routines to access the Finder Information.

```
#include "MacTypes.h"
#include "HFS.h"
#include "FileMgr.h"
#include "SegmentLdr.h"
#include "MemoryMgr.h"
#include "ToolboxUtil.h"

main()
{
   int*ptr, i;
   char setParms;
   Str255volName, s;
   AppFile*p;
```

```
   union
   {
      unsigned char c[4];
      OSType t;
   }
   theType;
   SetZone(SysZone);
   HUnlock(AppParmHandle);
   SetHandleSize( AppParmHandle,sizeof(AppFile)+4 );
   if (MemError())
      ReallocHandle( AppParmHandle, sizeof(AppFile)+4 );
   setParms = !MemError();
   HLock(AppParmHandle);
   SetZone(ApplZone);
   if (setParms)
   {
      p=(AppFile *)(*AppParmHandle+4);
      GetVol(volName,p->vRefNum);
      GetIndString(&s,100, 3);
      for (i=0;i<4;i++) theType.c[i] = s[i+1];
      p->fType = theType.t;
      p->versNum = 0;
      GetIndString(&s,100,2);
      BlockMove(&s,&(p->fName),s[0]+1);
      ptr = (int *)*AppParmHandle;
      *(ptr++) = 0;
      *ptr = 1;
   }
   theLaunch:
   GetIndString(&s,100,1);
   Launch(0,s);
}
```

•••

From: 6600pete@hub.UUCP
Subject: Re: Have you had problems with SetClikLoop?

From article <921@excelan.COM>, by mahboud@kinetics.com (Mahboud Zabetian):

> Hi. Anybody out there ever try calling SetClikLoop twice consecutively for
> different TEHandles? I am doing so and it seems that the second call also
> changes the value of the clikLoop field of the first TE! Sound strange?

Yes and no.

Yes, it's strange that it happens. Dunno why they chose to do it this way.

No, it's not strange in that I've seen it before.

My own situation was that a TE application I was writing set a ClikLoop, but a TE DA also wanted to do it. Under a certain set of circumstances, the DA would try to call its clikLoop and die because by that time its clikLoop pointed into outer space.

My solution was to SetClikLoop every time any of my TE windows became active. I also had to do some mucking with an (undocumented?) application global. I got this info from Michael Kahl, who apparently wrote a substantial portion of TE (if I read him correctly). Here's a code fragment in Pascal:

```
[ PROCEDURE Activate ( ... ); ]
[ VAR TEMagic : LONGINT; TEMagicPtr : ^LONGINT; ]

FUNCTION A5: longint;
    INLINE
        $2E8D;
```

```
        [ BEGIN... ]

            TEMagicPtr := pointer(A5 + 20);
            TEMagic := TEMagicPtr^;
            SetClikLoop(@myClikLoop, TEHandle(WindowPeek(whichWindow)^.refCon));
            TEClick(theEvent.where, BitAnd(theEvent.modifiers, shiftKey) <> 0,
                      TEHandle(WindowPeek(whichWindow)^.refCon));
            TEMagicPtr^ := TEMagic;

        [ ...END; ]
```

I wager this would be a lot more elegant, not to mention easier, in C.
-
Pete Gontier

●●●

From: jeff@tc.fluke.COM (Jeff Stearns)
Subject: Re: need Apple-Double format help

In article <SCOTTH.90Jan3150218@harlie.corp.sgi.com> scotth@sgi.com (Scott Henry) writes:

> ... Where can I get the structure of the resource fork of Apple-Double
> format? I have been able to reverse engineer parts of it ...

AppleSingle and AppleDouble specs are available from Apple or Cayman. I'll dig them up and post them myself in a few days if they don't show up in somebody else's posting by then.

Creating the resource fork isn't totally sufficient to make a double-clickable file, since the GatorBox stores some needed information in the .DESKTOP file. Creating a file behind the GatorBox's back won't result in a document that's directly launchable until the GatorBox knows what you've done. I don't know the exact details of what's necessary to make it work. Certainly rebuilding the desktop is sufficient; maybe unmounting and remounting the volume. Maybe just closing and reopening the folder.

To the fellow who wants to write a program to convert zmodem <-> AppleDouble <-> macbinary, here's a start I hacked out recently:

```
::::::::::::::
macbintogator
::::::::::::::
#! /bin/sh
#
#  Convert a MacBinary three-pronged file into an AppleDouble file suitable
#  for a GatorBox.  The .info file is retained in case a subsequent
#  reconversion is attempted.
#
#  The pathnames should be in "basename" form, without any trailing .info or
#  .rsrc or .data extension.
#
#   Jeff Stearns          jeff@tc.fluke.COM
#   John Fluke Mfg. Co, Inc. (206) 356-5064

for file do
    (
    : do nothing with $file.info &&
    mv $file.rsrc `dirname $file`/%`basename $file` &&
    mv $file.data $file
    ) || exit 1
done

exit 0
```

```
::::::::::::::
macbintoz
::::::::::::::
#! /bin/sh
#
#  Convert three UNIX files representing the 3 MacBinary forks
#  into one Zmodem file.
#
#  The three forks are concatenated as info, data, rsrc.
#
#  Each fork begins on a 128-byte boundary.  Holes are padded
#  with zeros.
#
#   Jeff Stearns          jeff@tc.fluke.COM
#   John Fluke Mfg. Co, Inc. (206) 356-5064

    (
    dd if=$1.info bs=128 conv=sync
    dd if=$1.data bs=128 conv=sync
    dd if=$1.rsrc bs=128 conv=sync
    )


::::::::::::::
ztomacbin
::::::::::::::
#! /bin/sh
#
#  Convert one Zmodem file into three UNIX files representing its 3 MacBinary
#  forks
#
#  The three forks produced are info, data, rsrc.
#
#  Within the info "fork" is encoded the length of the data and
#  rsrc forks:
#
#       bytes 53-56 = data length
#       bytes 57-60 = rsrc length
#
#  Each fork begins on a 128-byte boundary; that leaves some
#  padding after the data and rsrc forks.
#
#  N.B. Our use of adb presumes that this host uses Macintosh byte order;
#  that's true for a Sun.
#
#  Byte positions are numbered BETWEEN bytes; byte 0 lies just to the left
#  of the first byte in the file.  Thus we call byte N what you'd get if you
#  did seek(N, ...); read( ...)
#
#   Jeff Stearns          jeff@tc.fluke.COM
#   John Fluke Mfg. Co, Inc. (206) 356-5064

dd if=$1 bs=128        count=1 of=$1.info

DataStart=128
DataLength=`echo '53?D' | adb $1.info | awk '{print $2}'`

RsrcStart=`expr 128 + '(' '(' $DataLength + 127 ')' / 128 '*' 128 ')'`
RsrcLength=`echo '57?D' | adb $1.info | awk '{print $2}'`

echo 1>&2 "DataLength=$DataLength  RsrcLength=$RsrcLength"

dd if=$1 ibs=1 skip=$DataStart count=$DataLength of=$1.data
dd if=$1 ibs=1 skip=$RsrcStart count=$RsrcLength of=$1.rsrc
```

```
::::::::::::::
ztogator
::::::::::::::
#! /bin/sh
#
#  Convert one Zmodem file into two UNIX files representing its rsrc and data
#  forks.
#
#  Within the zmodem info "fork" is encoded the length of the data and
#  rsrc forks:
#
#       bytes 53-56 = data length
#       bytes 57-60 = rsrc length
#
#  Each fork begins on a 128-byte boundary; that leaves some
#  padding after the data and rsrc forks.
#
#  N.B. Our use of adb presumes that this host uses Macintosh byte order;
#  that's true for a Sun.
#
#  Byte positions are numbered BETWEEN bytes; byte 0 lies just to the left
#  of the first byte in the file.  Thus we call byte N what you'd get if you
#  did seek(N, ...); read( ...)
#
#   Jeff Stearns          jeff@tc.fluke.COM
#   John Fluke Mfg. Co, Inc. (206) 356-5064

Backup=$1.original

echo "To avoid overwriting your file $1, a backup copy is made at $Backup"
cp $1 $Backup

dd if=$Backup bs=128 count=1 of=$1.info 2>&-

DataStart=128
DataLength=`echo '53?D' | adb $1.info | awk '{print $2}'`

RsrcStart=`expr 128 + '(' '(' $DataLength + 127 ')' / 128 '*' 128 ')'`
RsrcLength=`echo '57?D' | adb $1.info | awk '{print $2}'`

rm $1.info

echo 1>&2 "DataLength=$DataLength  RsrcLength=$RsrcLength"

dd if=$Backup ibs=1 skip=$RsrcStart count=$RsrcLength of=`dirname
$1`/%`basename $1`
dd if=$Backup ibs=1 skip=$DataStart count=$DataLength of=$1


::::::::::::::
gatortoz
::::::::::::::
#! /bin/sh
#
#  gatortoz - Convert an AppleDouble file (actually a file and %file pair)
#             to a file in zmodem format.
#
#  Within the zmodem info "fork" is encoded the length of the data and
#  rsrc forks:
#
#       bytes 53-56 = data length
#       bytes 57-60 = rsrc length
#
#  Each fork begins on a 128-byte boundary.
```

```
#
#  N.B. Our use of adb presumes that this host uses Macintosh byte order;
#  that's true for a Sun.
#
#  Byte positions are numbered BETWEEN bytes; byte 0 lies just to the left
#  of the first byte in the file.  Thus we call byte N what you'd get if you
#  did seek(N, ...); read( ...)
#
#   Jeff Stearns          jeff@tc.fluke.COM
#   John Fluke Mfg. Co, Inc. (206) 356-5064


for file do
    #
    #  The data fork is found in file.
    #  The resource fork is found in %file.
    #  A temporary info fork is synthesized in /tmp/file.info.
    #
    Data="$1"
    Rsrc="`dirname $1`/%`basename $1`"
    TempInfo="/tmp/`basename $1`.info"

    MacFileName="$Data"
    MacFileNameLength=`echo -n "$MacFileName" | wc -c`
    DataLength=`ls -l $Data | awk '{print $4}'`
    RsrcLength=`ls -l $Rsrc | awk '{print $4}'`

    echo 1>&2 "$file:   Data length = $DataLength   Resource length = $RsrcLength"

    #
    #  Create the info fork as size 128 bytes.
    #  Insert the Macintosh filename at byte 2.
    #  In a minute, we'll use adb to patch in the byte count that
    #  must precede the filename (these are Pascal-type strings).
    #
    echo -n "XX$MacFileName" | dd bs=128 conv=sync of=$TempInfo 2>&-
    (
        echo "0?w $MacFileNameLength"
        echo "53?W $DataLength"
        echo "57?W $RsrcLength"
    ) | adb -w $TempInfo 1>&-

    #
    #  Each fork begins on a 128-byte boundary.  Gaps are NULL-padded:
    #
    dd if=$TempInfo bs=128 conv=sync 2>&-
    dd if=$Data     bs=128 conv=sync 2>&-
    dd if=$Rsrc     bs=128 conv=sync 2>&-

    rm $TempInfo
done
```

--
Jeff Stearns

●●●

From: zben@umd5.umd.edu (Ben Cranston)
Subject: Re:Adding a document to the launch of an application? (desktop file notes)
Summary: Corrects previous mistake

In article <5872@umd5.umd.edu> zben@umd5.umd.edu (Ben Cranston) blathers:

> One of the three words in the APPL resource is a link to the BNDL resource,
> which then has links to the ICN# and FREF resources.

I must have been high on fatigue poisons while writing that. I meant that the third word of the creator-ID resource contains the link to the bundle.

Looks like all you have to do is open a WD on the directory ID in the APPL resource, then set that as the current volume, then do a Launch on the name in the APPL resource. The following information on how to find the ICON is included here for reference:

Desktop File Notes:

The APPL resource contains triples of creator-ID, directory number, and string filename for each application present on the desktop's volume.

For example, MacWrite's creator-ID is MACA. Somewhere in the APPL resource there will be a record:

MACA 20 MacWrite 5.01

This tells the finder to SetWD to a WD on directory ID 20 and then Launch MacWrite 5.01 in order to execute the application.

The desktop file also contains a resource with the creator-ID as its resource ID (the resource number is always zero). The first four bytes seem to be constant (77231199?) but the last two bytes contain a link to the Finder's copy of the application's bundle resource. For example:

```
MACA(0)  7723 1199 109D    w#DoDu
              ----
```

Hexadecimal 109D is decimal 4253. Sure enough, there is a bundle resource with that number, that is a (relocated) copy of MacWrite's bundle:

```
BNDL(4253)
ICN#
1   7453
2   23840
3   15699
4   12586
FREF
1   28330
2   9671
3   29598
4   32610
```

These are the relocated ICN# and FREF resources as they appear in the desktop file. For example:

```
FREF(9671)    WORD     2
```

This says a document of creator-ID MACA and file type WORD should appear as relative icon number two. Tracking back through the bundle, we find that the appropriate icon is stored in ICN# 23840 in the desktop file.

Gee, thanks for mentioning it. I didn't even KNOW there was an APPL resource till you asked about it!

--
Ben Cranston <zben@Trantor.UMD.EDU>

●●●

From: amanda@mermaid.intercon.com (Amanda Walker)
Subject:**Re: Accessing Nubus board name**

In article <7492@pt.cs.cmu.edu>, mkb@rover.ri.cmu.edu (Mike Blackwell) writes:

> For a given Nubus slot, how do I access the name of the board (as stored in
> the configuration ROM)? I would guess it's in a slot resource, accessible by
> SGetCString(), but I don't have Designing Cards and Drivers so I don't know

> which one or where (and IM5 is no help). If you have a code fragment, or
> even a hint, I would be most obliged.

Here's my standard Slot Manager demo program. It compiles into an MPW tool (MPW C 3.0):

```
#include <stdio.h>
#include <slots.h>
#include <ROMDefs.h>

main()
{
    int i;
    SpBlock spb;
    long l;

    for (i = 9; i < 16; i++) {
        spb.spSlot = i;
        spb.spID = 1;
        spb.spExtDev = 0;
        if (!SRsrcInfo(&spb)) {
            printf("Slot %d:\n", i);
            spb.spID = sRsrcName;
            if (!SGetCString(&spb)) {
                printf("  %s\n", (char *) spb.spResult);
            }
            spb.spID = vendorInfo;
            if (!SFindStruct(&spb)) {
                spb.spID = vendorId;
                if (!SGetCString(&spb)) {
                    printf("  Vendor: %s\n",
                        (char *) spb.spResult);
                }
                spb.spID = serialNum;
                if (!SGetCString(&spb)) {
                    printf("  Serial Number: %s\n",
                        (char *) spb.spResult);
                }
                spb.spID = revLevel;
                if (!SGetCString(&spb)) {
                    printf("  Revision: %s\n",
                        (char *) spb.spResult);
                }
                spb.spID = partNum;
                if (!SGetCString(&spb)) {
                    printf("  Part Number: %s\n",
                        (char *) spb.spResult);
                }
                spb.spID = date;
                if (!SGetCString(&spb)) {
                    printf("  Revision Date: %s\n",
                        (char *) spb.spResult);
                }
            }
        }
    }
}
```

Enjoy,

--
Amanda Walker InterCon Systems Corporation

●●●

From: loganj@yvax.byu.edu

Subject: Re: For GURU'S Only (Starting Point for serial comm)

You can anonymous ftp a simple C program (source included) called "Capture" that demonstrates Macintosh serial port control from "noc.byu.edu" (128.187.7.2). Get a file in the "pub" directory called "CaptureSit.hqx". It includes a "LightSpeed 3.x" C project, source code, and resource file.

The program allows you to select either the printer port or the modem port and change the communication parameters for either port (baud rate, stop bits, parity, handshaking, ...) in a slick little dialog.

It is intended ONLY as a starting point for students and others that need to use Macintosh for serial communications. It acts like a very simple (and VERY slow) terminal emulator by displaying characters from the serial port in a TextEdit window. You're expected to make the necessary modifications to satisfy your communications requirements. One of the first changes is usually speeding up the communications by removing the TextEdit display capability. TextEdit is used initially only so students can "see" their data (if you know what I mean) and better understand how the program works.

Be sure to read the comments at the beginning of the main program,"Capture.c".

(I am not a GURU)

Regards,
Jim Logan loganj@byuvax.bitnet

●●●

From: Amanda Walker
Subject: Info about KeyCodes

In article <1990Feb6.205957.27722@ux1.cso.uiuc.edu>, dorner@pequod.cso.uiuc.edu (Steve Dorner) writes:
> It would be nice to hear that there are magic character codes for these keys,

What you have to do is look at the "keyCode" part of the event message instead of the "charCode" part. The function & editing cluster keys don't have unique ASCII codes, but they do have unique (and standardized) virtual key codes. IM V has a chart of (for example) the ADB Extended keyboard that shows all of the virtual key codes for the function keys etc.

Hope this helps,
--
Amanda Walker InterCon Systems Corporation

●●●

From: cbm@well.UUCP (Chris Muir)
Subject: Re: Modifying the mouse input (SetMouse code in pascal)
Summary: Here's the compliment to GetMouse.

Here's the other side of GetMouse, SetMouse:

```
procedure SetMouse (where: point);
  var
   LowGlob: integer;
   LowMem: ptr;
   PointPtr: ^point;
   finalTicks: longint;
  const
    {some dangerous low-memory-global equates}
   MBState = $172;      {byte}
   MTemp = $828;        {point}
   RawMouse = $82c;     {point}
   Mouse = $830;        {point}
   CrsrNew = $8ce;      {byte}
   CrsrCouple = $8cf;   {byte}
   Couple = $ff;        {value for CrsrCouple}
   Uncouple = $00;      {value for CrsrCouple}
 begin
```

```
  LocalToGlobal(where);             {Get ready to restore old mouse position}
  LowMem := pointer(RawMouse);      {point to low memory}
  PointPtr := @LowMem^;             {treat it as a point}
  PointPtr^ := where;               {store saved mouse position into it}
  LowMem := pointer(MTemp);         {point to low memory}
  PointPtr := @LowMem^;             {treat it as a point}
  PointPtr^ := where;               {store saved mouse position into it}
  LowMem := pointer(CrsrNew);
  LowMem^ := $ffff;                 {both CrsrNew & CrsrCouple}
  Delay(5, finalTicks);             {let the cursor catch up}
end;       {SetMouse}
```

Note that it uses undocumented low memory globals.

•••

From: lsr@Apple.COM (Larry Rosenstein)
Subject: Re: (Novice) help with Undo function

In article <4796@thor.acc.stolaf.edu> sobiloff@thor.acc.stolaf.edu (Blake Sobiloff) writes:

> > to write a program in Microsoft's QuickBASIC and I would like to impliment
> > a simple Undo feature. None of the sample code that comes with QB shows
> > an Undo implimentation. I thought maybe FracApp would show me, but Undo
> > seems to be handled by MacApp (gee, if I only had MacApp...). Soooo...

Traditionally, Undo is something people add as an afterthought. It is largely application-specific, which is why most sample programs don't do Undo.

I'll divide the problem into the user and programmer view.

From the user's view, there are several things to do.

(1) You should insert the name of the operation into the Undo item (so it reads Undo Paste or Undo Style Change, etc.) Also, if the user chooses Undo then you should change the menu item to read Redo Paste, etc. This ensures that the user can look at the item and see what it is going to do.

(2) When undoing a command, you should restore the select to the state it was immediately before the command was done. When redoing a command you should restore the selection to the state it had immediately after executing the original command. It is also desirable to scroll some part of the new selection into view, so the user can see what happened.

(3) You should implement Undo for all commands that change the document. Some programs don't allow you to undo commands such as type style changes, either because those are hard to implement or because the user can simply change the type style back to what she wants. The problem is that this is inconsistent, and the user can't tell if a certain command is going to be undoable.

(4) When the user performs the next command, the previous one is no longer undoable. (Except that some programs now support multi-level undo, in which this is not true.) It is nice if the user can undo a command even after saving a document, but that may be more difficult to implement. Also, if you support >1 document opened at once, you have to decide whether each document has its own undoable command.

On the implementation side, the obvious way to implement Undo is to save enough state to reverse the command and redo it if necessary. This works for most commands, but the exact implementation is application-specific.

In a text processor, any kind of typing command involves replacing one block of text with another block. (Either block may be empty.) In a bitmap editor, you replace one area with another. When moving an object, you simply remember how far you moved it, and move it in the opposite direction.

There are commands, however, for which this implementation takes too much memory. Consider changing the font in a text document. You would have to remember all the old font changes. Or consider changing the fill pattern for all objects in a drawing; you would have to remember all the original fill patterns.

The solution is to use a filtering approach in these cases. When the user changes the font, you don't change your internal data structures. Instead you remember the affected selection and the fact that he font has changed.
When it comes time to draw the text, you draw the affected text in the new font rather than the font recorded in the data structures.

To undo this command, you simply "remove" the filter and draw the text in its "true" font. When the command is no longer undoable, you make the change permanent by changing the data structures.

In either implementation, you need to remember the current selection at the time the command is executed. That's because the user can change the selection and still undo the previous command. (Changing the selection is not considered a change to the document.)

Larry Rosenstein, Apple Computer, Inc.

•••

From: Matthew Xavier Mora mxmora@unix.sri.com
Subject: Re:What is the format of the scrapbook file

I don't really know the format of the scrapbook file, but if its just PICT's that you want, that's easy. If look at the scrapbook file with resedit you see that it has a bunch of resources, PICT being one of them. If you open the PICT resource you can see a list of them. To get these out without knowing there Id numbers, you can use the rom call Get1IndResource (GetIndResource if you have 64K roms). This will give you a handle to the PICTs. Count1Resources will return the number of the resource type that you specified. Loop and call Get1IndResource each time through the loop to get a handle to the next pict in the file.

Below is some Ugly pseudo pseudocode.

```
ref:=OpenResFile('scrapbook');      {open scrapbookfile}
index:=Count1Resources('PICT');      {find out how many there are}
for i=1 to index do
    begin
        myPicthandle:=Get1IndResource(rType, index);  {get the next PICT}
        {check for nil}
        {do your conversion}

        DisposHandle(myPicthandle);

    end;

CloseResFile(ref);
end.
```

•••

From: svc@well.UUCP (Leonard Rosenthol)
Subject: Re: Programming the SCC (Code to poll the Macrecorder)
Summary: Here is some code...

In article <347@eldritch.hss.bu.oz>, grue@melmac.hss.bu.oz (Frobozz) writes:

> In article <YYXJcay00UoL89Vkp3@andrew.cmu.edu> nf0i+@andrew.cmu.edu (Norman William Franke, III) writes:
> >I would also like information on using the serial ports at speeds greater
> >than 57K, or more to the point, how to read data from a MacRecorder.
> >
> >Norman Franke
> >nf0i+@andrew.cmu.edu
>
>
> Count me in too. (esp the bit about a MacRecorder)
>
> Paul Dale

What follows is some source for reading the data direct from a Mac- Recorder that was posted to this group previously. I did not write it, so I take no credit and especially NO BLAME!

/* Written 6:18 pm Nov 26, 1988 by palmer@tybalt.caltech.edu in uxe.cso.uiuc.edu:comp.sys.mac.programmer */
Due to the large number of requests that have been posted recently, I am posting this code which reads the MacRecorder (sampling at 22 kHz).

MacRecorder comes with no programming documentation, so this is as good as it gets. If anyone has anything better, I'd be very interested.

I don't know where the original code comes from, but I made modifications which allows it to work as an oscilloscope (on an SE or earlier machine.) The style of the original code was lousy, and I opted to maintain stylistic unity with my additions.

The core of the program is the routines which set up the port for reading an externally clocked data stream. To understand these, you must read the Zilog SCC (Serial communications controller) data sheets. At the MacRecorder's input rate, interrupt driven routines are infeasible, so it uses continuous polling. This may cause data loss if other interrupts occur.

MacRecorder also transmits its data MSB first (if you understand how an SAR ADC works, you will understand why.) This is the opposite of most serial ports, and so a translation table (included) which reverse the bit order is needed.

Here is the code:

```
#include   <stdio.h>
#include   <QuickDraw.h>
#define    x1Clock        12  /* Clock mode for Scc chip */
#define    x16Clock   76


#define LENGTH 750        /* how much to read */
#define WIDTH 512     /* how large the screen is */
int Length = LENGTH,RealLength,x;
unsigned char buffer[LENGTH];
unsigned char table[256];
int screentable[256];
#define TOPMAR 20

void Die();

unsigned char *FindMin();

extern long *zero:0;
main()
{
    register unsigned char *pch;
    Rect r;
    long SccIn();

    int i;

    InitGraf(&thePort);
    InitFonts ();
    InitWindows ();
    InitMenus ();
    TEInit ();
    InitDialogs(&Die);
    InitCursor ();

    SccInit();
    TableInit();
    r = thePort->portBits.bounds;
    PaintRect(&r);

    r.top = TOPMAR;
```

```
    r.bottom = TOPMAR+256;

    for (i = 0 ; i < 256 ; i++)
        screentable[i] = (table[i] + TOPMAR) * thePort->portBits.rowBytes;
    do {
        PenNormal();
        RealLength = MySccIn(buffer, LENGTH);
        pch = FindMin(buffer, LENGTH - WIDTH);
        DrawCurve(pch);
        if (Button()) {
            SysBeep(3);
            Die();
        }
        PaintRect(&r);
    } while (RealLength>=LENGTH);
    SysBeep(12);
}

DrawCurve(pch)        /* works only on MacI under non-multi finder */
register unsigned char *pch;
{
    register int byte;
    register int bit;
    register int *pscreentable = screentable;
    register unsigned char *pscreen  =
               (unsigned char *)(thePort->portBits.baseAddr);

    for (byte = 0 ; byte < 64 ; byte++)
        for (bit = 0x100 ; bit >>= 1 ; )
            pscreen[pscreentable[*pch++] + byte] ^= (char)bit;
}

unsigned char *FindMin(pch, cch)
unsigned char *pch;
int cch;
{
    unsigned char *pchmin = pch;
    unsigned char min = *pch;

if (cch < 0) SysBeep(3);
    for ( ; --cch > 0 ; )
        if (*pch++ < min) {
            pchmin = pch - 1;
            min = *pchmin;
        }
    return pchmin;
}

char **SccRd,**SccWr,*SccRBase,*SccWBase;  /* pointer to Scc chip */
int aCtl = 2;                                       /* offsets */
int aData = 6;
int bCtl = 0;
int dData = 4;
SccInit() /* initializes the Scc chip for the MacRecorder Plus */
    /* for the regular MacRecorder II, replace x1Clock with x16Clock below */
{
    SccRd = (char **)0x1D8;
    SccWr = (char **)0x1DC;
    SccRBase = (char *)*SccRd;
    SccWBase = (char *)*SccWr;
    SccPoke(9,2);         /* NV only */
                        /* following line is for MacRecorder Plus */
    SccPoke(4,x1Clock);   /* 2 stop bits, Async, x1 clock mode  */
    SccPoke(1,1);         /* no Rx/Tx Int, Ext Int ON (mouse) */
    SccPoke(3,193);           /* initialize receiver, 8bits */
    SccPoke(5,122);           /* 8bits/char, send break(for other hardware!), Tx enable */
```

```
    SccPoke(11,48);          /* use TRxC as receiver clock */
    SccPoke(14,1);           /* BR enable, nothing else */
    SccPoke(15,8);           /* Interrupt on CD changes (mouse), turn off CTS interrupt */
    SccPoke(64,64);          /* Reset Rx CRC */
    SccPoke(9,10);           /* initialize master interrupt and NV */
}
SccPoke(n,v)      /* accesses the modem port */
char n,v;
{
    *(SccWBase + aCtl) = n;  /* set index to register n */
    *(SccWBase + aCtl) = v;  /* write v into register n */
}
SccPeek(n)
char n;
{
    *(SccWBase + aCtl) = n;          /* set index to register n */
    return(*(SccRBase + aCtl));      /* retrun value in register n */
}
TableInit()
{
    MakeTable(&table[0]);
    ModifyTable(&table[0]);
}
MakeTable(table)
register char *table;
{
    asm     {
            adda.w #256,table
            move.w #255,D0
    lp0:    move.b D0,D1
            move.w #07,D3
    lp1:    lsr.b  #01,D1
            roxl.b #01,D2
            dbf        D3,@lp1
            move.b D2,-(table)
            dbf        D0,@lp0
        }
}
#ifdef FOO   /*( this is not needed if you use unsigned chars */
ModifyTable(table)
char *table;
{
    int i;
    for (i=0;i<256;i++)
    {
        if ( table[i]>=0 ) table[i] -= 128;
        else table[i] += 128 ;
    }
}
#endif
int MySccIn(dest, count)
register char *dest;
register int count;
{
    int     iCount = count;   /* actual number of bytes received */
    register   char    *pSccRBase = SccRBase;
    while (count-- > 0 && !Button()) {
        while (0 == (pSccRBase[2] & 0x1))
            if (Button())
                return (iCount - count + 1);
        *dest++ = pSccRBase[6];
    }
    return iCount;
}

long SccIn(dest,count)
```

```
register char *dest;
register long count;
{
    register   unsigned char  *Table;              /* a lookup table */
    register   long   aCount;       /* actual number of bytes received */
    Table = &table[0];
    aCount = 0L;
    asm    {
        move.l  #0x9FFFF8,A0               ; /* SccRBase   */
#ifdef FOO
        move.l #0xEFE1FE,A1                ; /* mouse button */
        clr.l  D0
    lp:    btst   #03,(A1)                        ; /* mouse clicked? */
        beq        @lq
#else
    }
    lp:    if (Button())
           return(aCount);
    asm {
#endif
        btst   #00,2(A0)                   ; /* SccRBase + aCtl */
        beq        @lp
        move.b 6(A0), D0                   ; /* SccRBase + aData */
        move.b 00(Table,D0), (dest)+       ; /* translate in lookup table */
        addq.l #1,aCount                   ; /* one more byte received */
        subq.l #1,count
        bne        @lp
    lq:    nop
        }
    return(aCount);
}

void Die()
{
    ExitToShell();
}
-----------------------------------------------------------------
```

David Palmer
palmer@tybalt.caltech.edu
...rutgers!cit-vax!tybalt.caltech.edu!palmer
"I was sad that I had no shirt, until I met a man with no torso"
/* End of text from uxe.cso.uiuc.edu:comp.sys.mac.programmer */

--

Leonard Rosenthol

•••

From: austing@Apple.COM (Glenn L. Austin)
Subject: Re: Width of popup menus
Keywords: How wide should they be?

mcdonald@fornax.UUCP (Ken Mcdonald) writes:

>Kinda wondering if anyone out there has come up with a nice solution
>to this problem . . .

>It's easy to set the size of the popup menu field in a dialog, when
>the popup menu is constant--just give it the width of the menu. But
>what about when things might be added to or removed from the menu. If
>the box as drawn in the dialog ends up wider than the menu, then clicking
>too far to the right pops up the menu, without actually having the cursor
>in the menu. The obvious (only, I guess) solution is to resize the box

>in the dialog, whenever the menu contents change. Is this an Apple-
>approved solution? Should I change the size of the box every time the
>user selects a new item? What a pain!

To quote Inside Mac:
“You can use CalcMenuSize to recalculate the horizontal and vertical dimensions of a menu whose contents have been changed (and store them in the appropriate fields of the menu record). CalcMenuSize is called internally by the Menu Manager after every AppendMenu, SetItem, SetItemIcon, and SetItemStyle call.”

Personally, if I'm changing a popup menu, I go ahead and redraw the box. Ugly, yes. Prettier than a popup appearing "inside" my box, however...

--
Glenn L. Austin

●●●

From: ccc_ldo@waikato.ac.nz (Lawrence D'Oliveiro, Waikato University)
Subject: Re: Script Manager Date Questions

I can't answer the rest of Robert's questions, but here's a slightly simpler
Long2Comp function:

```
Function Long2Comp
  (
TheLong : LongInt
  ) : Comp;
  { converts an unsigned long integer to a Comp. }

Var
    Result :
    Record
        High, Low : LongInt
    End {Record};

  Begin
Result.High := 0;
Result.Low := TheLong;
Long2Comp := Comp(Result)
  End {Long2Comp};
```

Lawrence D'Oliveiro

●●●

From: rdclark@Apple.COM (Richard Clark)
Subject: Re: Changing Radio Button Titles

rk39+@andrew.cmu.edu (Robert Joseph Kuszewski) writes:

>Does anyone know how to change the title of a radio button within a program?
>Bob --

Try:

```
PROCEDURE SetCTitle(ctl: ControlHandle; title: STR255);
```

in the control manager.

If the button is in a dialog, you can use GetDItem to get the handle to the control. If it's in a window, your best bet is to store the ControlHandle somewhere at the time you create it.

Richard Clark

●●●

From: zben@umd5.umd.edu (Ben Cranston)
Subject: Re: Leading dashes in menu items.
Summary: Put a null as the first byte of the string

In article <573@argosy.UUCP> freeman@cleo.UUCP (Jay R. Freeman) writes:

> I need to create a menu item whose entire text consists merely of a
> single minus sign. Of course I can't just do that, because a single
> minus sign is menu-manager meta-syntax for drawing a dashed line
> across the menu. And it appears that a leading minus sign is NOT one
> of the meta-characters whose effect you can get around by using
> SetItem (I think it is) on the actual text.

The trick I have always used in this case is to make the first character of the menu item text a null (character code zero). This is not displayed but is enough to thwart the comparison in the menu manager. The combination of this and always using SetItem or whatever seems to work for me:

```
string[0] = 2;
string[1] = 0;
string[2] = '-';
SetItem(blatmenu,BMDASH,string);
```

Coding from memory without my trusty IM-I so don't flame if I goofed. Cannot remember where the null trick came from, might be IM, tech note, the network, MacTutor, who knows...

Lest we belabor the obvious, make sure this doesn't faze any of your code that might do a GetItem and look at the string, for you'll surely see the null come right back to you!

•••

From: aries@rhi.hi.is (Reynir Hugason)
Subject: Re: Str255 --> 'STR ' how?

In <1990Jun12.174107.29196@ucselx.sdsu.edu> purcell@sciences.sdsu.edu (Guy B. Purcell) writes:

>I know this is probably soooo simple, but... How do I save the contents of a
>Str255 string in a 'STR ' rsrc? I'm saving config settings, and have managed to
>save them if the new string is <= the old one in length (char-by-char). Is
>there some standard method of doing this (non-char-by-char so length is not a
>factor)? Thanks.

It sure is simple ;-)

Try:

```
PROCEDURE SaveStrAsSTR(whichStr: Str255; resID: INTEGER);
  VAR
    strHdl: StringHandle;
    resHdl: Handle;
  BEGIN
    strHdl:=NewString(whichStr);
    IF (strHdl = NIL) THEN EXIT(SaveStrAsSTR);

    resHdl:=GetResource('STR ', resID);           { Check if res exists  }
    IF (resHdl <> NIL) THEN               { rmve it if it does   }
      BEGIN
        RmveResource(resHdl); DisposHandle(resHdl);
      END;
    AddResource(Handle(strHdl), 'STR ', resID, '');  { now add'em }
    WriteResource(Handle(strHdl));
    ReleaseResource(Handle(strHdl));
  END;
```

> Guy (purcell@zeus.sdsu.edu)

+++
Mimir (aries@rhi.hi.is) - Aries, Inc.

●●●

From: austing@Apple.COM (Glenn L. Austin)
Subject: Re: Where are disk icons kept?

jjoshua@topaz.rutgers.edu (Jon Joshua) writes:

> >Here's a tricky one....
> > I want to write a function that takes a SCSI device offline.
> >I know about _OffLine but the damn thing doesn't gray out the icon
> >(a-la the floppy disk). It looks as though I'll have to do this by
> >hand.
>
> >Where in memory is the drive's icon?

You can get the info via a control call to the driver with a csCode of 21 (dec) and will get back an ICN# and Str255 (in that order) in a pointer passed back in csCode with noErr. Drive nbr goes in ioDrvNum.

--
Glenn L. Austin

●●●

From: lsr@Apple.COM (Larry Rosenstein)
Subject: Re: How to tell when an appl quits?

In article <1990Jun18.184859.5031@ncsuvx.ncsu.edu> hench@csclea.ncsu.edu (Steven Hench) writes:
> > to tell when it's done. For example, suppose you wanted an
> > init to automatically switch to B&W when you run Colony,

One question is whether you want to notice when the user switches out of Colony (or maybe that's not possible?)

Someone asked me this question last week, and the only thing I could come up with off hand was patching CloseResFile. The problem is filtering out the cases where the file is closed normally from those when the application is quitting.

You could take the refnum and use PBGetFCBInfo on it to get the file name and compare that with CurAppName. If they match then the application is quitting. This filters out cases where an application is closed by (say) ResEdit. Also, under MultiFinder the Finder opens and closes the application before launch, presumably to get the SIZE resource; but CurAppName is still Finder at those times.

Larry Rosenstein, Apple Computer, Inc.

●●●

From: KOFOID@cc.utah.edu
Subject: Re: How to get the current application (with code)

Am I missing something in this thread? I've used the following function for years and it's always worked:

```
FUNCTION Self(): Str255;

    VAR
        apName : Str255;
        apRefNum : Integer;
        apParam : Handle;

BEGIN
    getAppParms(apName, apRefNum, apParam);
    Self := apName
```

```
    END; {Self}
```

This returns the name of the application running the code. If the question is "What is the current active application under MultiFinder?", then SELF will probably *not* provide the correct answer. I haven't tried it under these conditions.

Cheers,

Eric.

•••

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: Random number Generator wanted.

kr@asacsg.mh.nl (Koos Remigius) writes:

```
.> This function gives me a random integer ( between 0 and range ).
.> function Randomize(range:Integer):Integer;
.> var
.>   rawResult:LongInt;                { "Raw" random number received from toolbox }
.>   begin  { Randomize }
.>     rawResult:=(ABS(Random);     { Get random number between 0 and 32767 }
.>     Randomize:=(rawResult * range) div 32768;{ Scale to specified range }
.>   end;   { Randomize }
```

Much more efficient:

```
Randomize:=abs(random) mod succ(range);
```

```
.>
.>What I need is a random number generator that gives me a random LongInt back.
.> The random number I want must lis between 0 and 2097151.
```

```
var Randomize:longint;
Randomize:=(32768*(abs(random) mod 64)+abs(random));
```

```
> Greetings from Holland,
> Koos Remigius.
```

Regards from Philadelphia.
--
Nick Jackiw I Visual Geometry Project I Math Department

•••

From: rhb@sc7.hgc.edu (Roger H. Brown)
Subject: Re: Random number Generator wanted.
Keywords: Random number

In article <152@asacsg.mh.nl> kr@asacsg.mh.nl (Koos Remigius) writes:

```
>
>What I need is a random number generator that gives me a random LongInt back.
>The random number I want must lis between 0 and 2097151.
>
>Koos Remigius.
```

If you can get your hands on a copy of the following article you will get the full details of generating uniform random integers between 1 and 2147483646 or up to maxInt = 2^31 -1.

See:
S.K. Park and K.W. Miller, "Random Number Generators: Good Ones are Hard To Find," Communications of the ACM, Vol.31, No. 10, 1988.

Until then, you might want to play with the following code:

```
  var seed : integer
```

```
  function Random : real
            (* Uses integer arith, yet returns a real between
                0.0 and 1.0  *)
  const
    a = 16807;
    m = 2147483647;
    q = 127773;   (* m div a *)
    r = 2836;     (* m mod a *)
  var
    lo, hi, test : integer;
  begin
    hi := seed div q;
    lo := seed mod q;
    test := a * lo - r * hi;
    if test > 0 then
      seed := test
    else
      seed := test + m;
    Random := seed / m
  end;
```

Regards

l Roger H. Brown GEnie: R.BROWN64 AOL: Roger48 l

•••

From: kaufman@Neon.Stanford.EDU (Marc T. Kaufman)
Subject: Re: Random number Generator wanted.
Keywords: Random number

In article <152@asacsg.mh.nl> kr@asacsg.mh.nl (Koos Remigius) writes:

> >What I need is a random number generator that gives me a random LongInt back.

If you have a Mac II, the following might help.:

```
***

        SEG         'Main'
*****************************************************************
*  ROUTINE         Random.a
*  FUNCTION        Provide Minimal-Standard random number generator
*  CALLING         C-compatible calling sequences
*
*  This generator is taken from CACM, October 1988, p.1192
*
*  It was designed by Stephen K. Park and Keith W. Miller
*
*  For future reference:  Possible better multipliers are 48271 and 69621
*****************************************************************
        CASE    ON
        MACHINE     MC68020

        PROC
        ENTRY   MSRandomSeed

Start   DC.W    ' ) Copyright Kaufman Research, 1988 '

MSRandomSeed
        DC.L    1       ; Place to store seed value

        ENDPROC
```

```
*****************************************************************
*  ROUTINE        MSGetSeed
*  FUNCTION       Return the current seed value
*  INPUT       none
*  OUTPUT      D0 = the seed
*****************************************************************
MSGetSeed  FUNC    EXPORT

       move.l MSRandomSeed,D0
       rts

       ENDFUNC

*****************************************************************
*  ROUTINE        MSRandom
*  FUNCTION       Return the next random number
*  INPUT       none
*  OUTPUT      D0 = the number
*****************************************************************
MSRandom   FUNC    EXPORT

       lea        MSRandomSeed,A0
       move.l (A0),D1
       mulu.l #16807,D0:D1          ; long multiply
       divu.l #$7fffffff,D0:D1      ; modulo 2^31 -1
       move.l D0,(A0)
       rts

       ENDFUNC

*****************************************************************
*  ROUTINE        MSRanSet
*  FUNCTION       Set the random number seed
*  INPUT       The seed value
*  OUTPUT      The seed value
*****************************************************************
MSRanSet   FUNC    EXPORT

       move.l 4(A7),D0              ; the value the user wants to use
       and.l  #$7fffffff,D0
       beq.s  MSGetSeed             ; zero is not a valid value
       cmp.l  #$7fffffff,D0
       beq.s  MSGetSeed             ; neither is 2^31-1
       lea        MSRandomSeed,A0
       move.l D0,(A0)
       rts

       ENDFUNC

       END
```

Marc Kaufman (kaufman@Neon.stanford.edu)

•••

From: davidd@ttidca.TTI.COM (David Dantowitz)
Subject: Re: Random number Generator wanted.
Keywords: Random number

Be quite careful how you splice together random numbers to form larger random numbers. Linear-conguential random number generators (of the form: seed = seed*a mod m + b) have more "random" high order bits than low order bits. Appending two 16 bit numbers may skew your application in a way that's difficult to measure.

If you want a 32 bit random number then check out Knuth for some multipliers (a) and modulos (m) that will give you "good" random numbers. Sorry I don't have it handy.

--
David Dantowitz

•••

From: kaufman@Neon.Stanford.EDU (Marc T. Kaufman)
Subject: Re: Random number Generator wanted.
Keywords: Random number

In article <1990Jun29.020739.9146@Neon.Stanford.EDU> kaufman@Neon.Stanford.EDU (Marc T. Kaufman) writes:
>In article <152@asacsg.mh.nl> kr@asacsg.mh.nl (Koos Remigius) writes:
->What I need is a random number generator that gives me a random LongInt back.

and I write back:
>If you have a Mac II, the following might help.:
[some assembly code for the Mac deleted]

I recently received the following mail:

"You may like to know that the Random function in the Mac ROM *is* the same as the "minimal std" function published in the CACM paper. It's just that it truncates the computed result to 16-bits. However, you can get at the 32-bit result by accessing the QD global "RandSeed". This works because Random uses the linear congruential method which is iterative, and saves the next computed random no in the sequence in RandSeed. Ie do something like (in C)

```
(void) Random();  /* ignore 16 bit result */
theValue = qd.randSeed; /* use 32-bit result */
```

In Pascal, just assign the unwanted 16-bit result to a dummy.

I compared 10000 rn's from a C version of the CACM rn generator with that generated by the ROM's Random() function, and they were identical.

Best regards

Sak Wathanasin

I thank Sak for this info. I am not sure just which Macs this applies to --probably the ci and fx, certainly not the SE. Can someone from Apple tell us when the algorithm changed, and is it part of a system patch so all Macs can benefit from it?

Marc Kaufman (kaufman@Neon.stanford.edu)

•••

From: nolan@tssi.UUCP (Michael Nolan)
Subject: Re: Random number Generator wanted.
Summary: Try any of the algorithms in Knuth, sample given
Keywords: Random number

It's fairly easy to write a random number generator, and there are several in Knuth's "Art of Computer Programming", I *think* in volume 2, at any rate the subject matter of the correct volume is algorithms.

One I've used many times involves three relatively prime numbers n1, n2, and n3. (Relatively prime means these numbers have no common factors above 1.)

Please pardon my pseudocode:

```
This_Random_Number = (Previous_Random_Number * n1 + n2) mod n3.
```

Of course, you need to provide an initial Previous_Random_Number as a seed.

This will generate a random number sequence of period n3. I assume this will work in longints.

Mike Nolan

•••

From: vladimir@prosper (Vladimir G. Ivanovic)
Subject: Re: Random number Generator wanted.
Summary: See CACM v31 #6 and v31 #10 for good, portable random number generators
Keywords: random number generator portable efficient correct

The June 1988 (v31 #6) issue of the Communications of the ACM has an article by Pierre L'Ecuyer called, "Efficient and Portable Combined Random Number Generators". The following October (v31 #10), Stephen Park and Keith Miller published "Random Number Generators: Good Ones are Hard to Find."

Both articles are ESSENTIAL reading for anyone interested in random number generation, and this includes naive users. The conclusion of both articles is that horrible random (sic) number generators are used by people who don't know better, while very good ones take less than 20 lines of Pascal! Amoung the horrid generators are some that come with certain systems or are presented in textbooks!

Here are the two proposed random number generators. Even though I have proof read them twice, I made no guarantees whatsoever about the correctness or the usability of the code below.

Here is the Minimal Standard proposed by Park and Miller:

```
function Random : real;
{ Initialize seed with 1..2147483646 }
{ maxint must be greater than 2**31 -1 }
const
  a = 16807;
  m = 2147483647;
  q = 12773;       { m div a }
  r = 2836;       { m mod a }
var
  lo, hi, test : integer;
begin
  hi := seed div q;
  lo := seed mod q;
  test := a * lo - r * hi;
  if test > 0 then
     seed := test
  else
     seed := test + m;
 Random := seed / m
   end;
```

Here is the Portable Combined Generator of Ecuyer for 32-bit computers. It has a period of roughly 8.12544 x 10**12.

```
Function Uniform : real;
{ Initialize s1 to 1..2147483562; s2 to 1..2147483398 }
  var
    Z, k : integer;
  begin
    k  := s1 div 53668;
    s1 := 40014 * (s1 - k * 53668) - k * 12211;
    if s1 < 0 then s1 := s1 + 2147483563;

    k  := s2 div 52774;
    s2 := 40692 * (s2 - k * 52774) - k * 3791;
    if s2 < 0 then s2 := s2 + 2147483399;

    Z := s1 - s2;
    if Z < 1 then Z := Z + 2147483562;

    Uniform := Z * 4.656613e-10;
  end;
```

All are urged to read both articles for a much better presentation. And you don't have to be a mathematician to understand them: very little of either article is not accessible to a competent programmer.

Enjoy!

-- Vladimir

Chapter 2 Code Resources (Inits,Cdevs,VBLs,...)

INITs and such... 38,39
How do you write a INIT in PASCAL? 39
Help! (jGNEFilter, GetNextEvent, events) 42
Info on Tail patches 42
Tail patches 43
Menu font list appearing as font (source to MDEF) 44
How can a CDEV "talk" to its INIT? 45
How do I install a driver with an INIT? (with Code) 46
cdev - INIT Data Exchange.....50
cdev - INIT Data Exchange.....51
CDEV/INIT Data Exchange, another way.....51
cdev - INIT Data Exchange.....52
CDEV/INIT Data Exchange, another way.....53
Informative INITs (code incl. 53
24-Hour FKEY? 58
VBL tasks Think C 4.0 59

From: russotto@eng.umd.edu (Matthew T. Russotto)
Subject: Re: INITs and such...
Keywords: mac INIT

In article <7429@ur-cc.UUCP> kellogg@prodigal.psych.rochester.edu.UUCP (Lars Ke:

> >Would some one please describe to me how to get an INIT to hang around in
> >memory and do something WITHOUT having to patch an OS routine?

Sure-- just DetachResource and HNoPurge yourself (you can depend on A0 being a handle to your code, or, more safely, get a pointer to the beginning of the code and RecoverHandle it) I prefer to instead copy my code to the system heap, that way I don't need to copy installation code.

> >For instance, continually check a specific folder for a specific file, and
> >notify the user if it appears - I know this has been done, but it serves as
> >an example. Can this be done without patching something?

I think so-- use asynchronous file manager calls, with a completion routine that re-queues the call if it failed (be sure to put a delay in though-- you may need a VBL task which re-queues the call to do this). Use the Notification manager to inform the user. (I think it's safe to do from completion routines)

> >On a slightly different subject, is it even possible to patch an OS routine
> >without using assembly language?

Sure, it can be done with stack-based routines with either C or PASCAL.

•••

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: INITs and such...

kellogg@prodigal.psych.rochester.edu.UUCP (Lars Kellogg-Stedman) writes:

> > Would some one please describe to me how to get an INIT to hang around in
> > memory and do something WITHOUT having to patch an OS routine?

You can get it to hang around all you want by putting it into the system heap, either by _BlockMoving the relevant code from your INIT resource into a _NewPtr,SYS block or by loading + detaching the resource directly into the heap. Unfortunately, while hanging around, it won't *do* anything, because no part of the regular functioning of the Macintosh has been instructed to execute it periodically.

While the most common method of ensuring your INIT's code gets executed is by patching a trap (especially a trap the invocation of which is related to the purpose of your code--e. g. patch _MenuSelect if you want to preprocess menu displays), if you need periodic invocation regardless of which traps are invoked by the user, you could consider installing a VBL task. These are called by hardware interrupts, and are discussed in Inside Mac vol I, Chapter 11, and in vol V, chapters 24 and 29. Be sure your VBL task is installed in the system heap.

Unfortunately, VBL tasks must assume a very restricted environment, and therefore, cannot perform certain operations (allocating and disposing memory, for instance). Conversely, trap-patches can operate in whatever environment the trap is documented as assuming, which--in some cases--is quite flexible.

> > For instance, continually check a specific folder for a specific file, and
> > notify the user if it appears - I know this has been done, but it serves as
> > an example. Can this be done without patching something?

I am the author of ChainMail, an INIT/cdev which polls a given folder for the existence of a specific file or the existence of any files. I designed ChainMail as a trap-patch (to _GetNextEvent) for the following reasons:

- Accessing the disk is difficult, if not impossible, from a VBL task.
- My task takes a noticeable amount of time. _GetNextEvent is not called in the middle of time-critical bits of code, and is generally assumed to mean that the application has little better to do than waste time.

> > On a slightly different subject, is it even possible to patch an OS routine
> > without using assembly language?

One is strongly advised against it. On the otherhand, patching a trap takes a *minimum* of assembly, and can be done in line. The actual code executed by the patch may be written in any language you chose, if your compiler can generate code resources.

> ~ ~ l Lars Kellogg-Stedman l "Software rots if not used"

Nick Jackiw jackiw@cs.swarthmore.edu "Just break out the

•••

From: Lawrence D'Oliveiro
Subject: Re: INITs and such...

Just to add my bit to everybody else's comments on this.

There are two mechanisms for queueing an operation to occur at a specific time: installing a VBL task, or putting an entry into the Time Manager queue. In both cases, your code would get called at interrupt level, so it can't do anything that would involve the memory manager (file and device I/O is OK).

If you need to do some operation involving allocating/deallocating/locking/ unlocking memory, you can queue it from your interrupt-level code for later execution via the Notification Manager. Remember, you don't actually need to post any notifications to the user if you don't want to: you can use the Notification Manager simply as a safe, clean way of queueing work to be done at Get/WaitNextEvent time.

Lawrence D'Oliveiro

•••

From:Charles Martin
Subject :Re:How do you write a INIT in PASCAL?

Isn't it amazing? No advice anywhere. Try this in THINK Pascal, the basic idea is to push the old trap address on the stack right before your patch by modifying the header that THINK Pascal puts on your CODE resource. If someone sees any grave errors or omissions (such as, what is the role of the 'sysz' resource, and does it matter that the address of the trap won't be in ToolScratch), please let me know!

(Of course there's a little implicit assembly here, it's inescapable!)

----- beep.p: code resource, CODE 128, locked, system heap, preload

```
unit beep;
interface
  procedure main;
implementation
  procedure main;
    begin
      SysBeep (1)
    end;
end.
```

----- install.p: code resource, INIT 0, locked; use resource file beep

```
unit install;
interface
  procedure main;
implementation
  const
    YourTrapGoesHere = $A9xx;
  type
    long_hdl = ^longptr;
    long_hdl = ^long_ptr;
    long_t = array [0..0] of longint;
  procedure main;
    var beep: Handle;
```

```
        long: long_hdl;
        addr: longint;
    begin
      beep := Get1Resource ('CODE', 128);
      DetachResource (beep);
      long := long_hdl (beep);
      addr := GetTrapAddress (YourTrapGoesHere);
      long^^ [5] := $2F3C; { MOVE.L <addr>,-(A7) }
      long^^ [6] := addr;
      SetTrapAddress (longint (beep^) + 22, YourTrapGoesHere)
    end;

end.
```

Charles Martin // martin@gargoyle.uchicago.edu
In-reply-to: mxmora@unix.SRI.COM (Matt Mora)

In article <11608@unix.SRI.COM>, mxmora@unix (Matt Mora) writes:

> >What I am really looking for is any examples in PASCAL.

Sorry, there were a couple of typos. The correct lines are:

```
    long_hdl = ^long_ptr;
      long^^ [5] := $2F3C; { MOVE.L <addr>,-(A7) }
      long^^ [6] := addr;
```

Charles Martin // martin@gargoyle.uchicago.edu

•••

From: Scott A. Mason
Subject: .c2.How to write a vbltask(with code)

In article <604@mit-amt.MEDIA.MIT.EDU> adam@mit-amt.MEDIA.MIT.EDU (Adam Glass) :

> >I've been having the darndest time trying to make a program to do a VInstall.
> >I can't seem to be able to get the PASCAL variant records/C unions to work.
> >Could some kind soul mail me a procedure/quickie program to do it? You could
> >also post it, as I'm sure other people will find it interesting.

Ok, here's a snippet of my code written in LSC. (It doesn't do much, but should help a lot of people getting things to work.)

```
#include <VRetraceMgr.h>

VBLTask vblTask;

void VBL_Routine ()
{
        SetUpA5();              /* set up for globals */
        vblTask.vblCount = 30;  /* reset the count so we run next time */
/* your code to do something real goes here */
        RestoreA5();            /* reset the globals stuff */
}

/* This routine installs the given VBLTask */

Set_Interrupts (vblTask)
VBLTask *vblTask;
{
        OSErr err;

        vblTask->qType          = vType;
        vblTask->vblAddr        = (ProcPtr) VBL_Routine;
```

```
        vblTask->vblCount       = 30;
        vblTask->vblPhase       = 0;
        err = VInstall ( vblTask );
}
```

Well, there it is in all its simplicity. I must say it was difficult the first time for me as well. The implementation of VBL tasks is not very clear in IM. Hope this helps.
--

Scott A. Mason

•••

From: sdh@flash.bellcore.com (Stephen D Hawley)
Subject: .c2.How to write WDEFs (with code)

I tried to mail this but it bounced. Here's what you need to do to write a WDEF:
Create a new THINK C Project, set the project type to be WDEF. Set the resource number to something nice for you.

The main should look like this:

```
pascal long main(varCode, theWindow, message, param)
short varCode;
WindowPtr theWindow;
short message;
long param;
{

    /*
     * This is probably the most straight forward approach
     * to implement the function.
     * return a long as appropriate for each part of the
     * window.
     */
    switch (message) {
    case wDraw:
        break;
    case wHit:
        break;
    case wCalcRgns:
        break;
    case wNew:
        break;
    case wGrow:
        break;
    case wDrawGIcon:
        break;
    }
}
```

Fill in the blanks as you need, and compile it into a WDEF with "Build Code Resource".

To debug, write a small project that creates a bunch of windows (using the resource number you specified before, allowing for variation codes etc) and lets you do all the playing with them. Then make sure you WDEF is put into a file called whatevermysmallprojectiscalled.rsrc where "whatevermysmallprojectiscalled" is the name of the small shell application.

When you want to start using it in a real program, use ResEdit to copy it into your project's resource file, or use the Merge option in Think C to merge it into your project.

Check out the Think C manual (4.0 p. 84-86) for more details.

Good Luck!
Steve Hawley

•••

From: cak3g@astsun7.astro.Virginia.EDU (Colin Klipsch)
Subject: Re: Help! (jGNEFilter, GetNextEvent, events)

In article <14463@reed.UUCP> chaffee@reed.UUCP (Alex Chaffee) writes:

> >In article <25ff4e70.643e@polyslo.CalPoly.EDU> rcfische@polyslo.CalPoly.EDU
> ><in the menu, even when the menu gets re-drawn. Then you might want
> ><to look at hooking into GetNextEvent so that you can intercept mouse
> ><downs and see if they're on your icon.
> >
> >A small suggestion - rather than patching GetNextEvent, take advantage of
> >the low-memory global jGNEFilter, which is sort of a built-in tail patch on
> >GNE. It's documented in a tech note (I can't remember the number); I could
> >give you source code if you're interested.
>
> ><Ray Fischer
> >--
> >Alex Chaffee

Tech Note #85 talks about jGNEFilter. It is a low-memory global which points to code that will be executed upon every event, after the Mac has done some of its own processing. I used to use this method of event filtering for TappyType, and it worked as follows. . .

In your INIT:

- Load your event-filter code, presumably stored as another resource, into the system heap.
- Make sure it's locked. (i.e. set its resLocked bit beforehand)
- Detach it with _DetachResource.
- Take the whatever pointer you find in jGNEfilter and store it somewhere safe in your event-filter.
- Store the address of your event-filter in jGNEfilter.

Now every time an event happens, your event-filter (tucked safely inside the system heap) will be called. Hopefully.

In your event-filter:

- Upon entry, A1 should point to the event record. Do whatever you want to do with the event.
- After you've done your thing, jump to the old value of jGNEfilter that you saved previously.

In effect, you are "splicing" your event-filter into the flow of event processing. If everyone were to do the same thing, then many different event filters could co-exist.

Two Bad Things: Unfortunately, not everyone plays nicely, and thus it may matter when your INIT runs in relation to others. Secondly, jGNEfilter is a low memory global, and thus it is not guaranteed to be around in the long run.

Patching GetNextEvent, as a method of event filtering, won't work except as a tail patch, which is a Thing Not To Be Done.

The scheme I use now is head-patching SystemEvent, which also seems to be called after every event. SystemEvent is "guaranteed" to be supported, and you don't have to share a precarious memory global with random strangers.

Colin Klipsch

•••

From: shebanow@Apple.COM (Andrew Shebanow)
Subject: Re: Info on Tail patches

Tail patches are trap patchs which do processing after calling the original trap, or modify the stack before calling the original patch. They're called tail patches because the typically look like this:

```
LEA    origTrapAddress,A0
; do preprocessing
JSR    (A0)
; do postprocessing
```

Since the bad part is at the end of the patch, its a tail patch.

A "clean" patch will have a format like this:

```
; do processing
LEA    origTrapAddr,A0
JMP    (A0)
```

Tail patching is bad because of the techniques that Apple uses in its System Software to fix bugs. Sometimes, a trap (lets call it _TrapA) has bugs in it which would be very difficult to fix without rewriting the trap's entire code. Rather than do that, Apple will sometimes patch a different trap (_TrapB) which is called by the broken trap (_TrapA). _TrapB will check the address of the caller on the stack and compare it to the hardcoded address of _TrapA's call, and, if the addresses match, behave in a different manner to fix the bug. This can cause a huge savings in speed and memory usage, but it prevents the use of tail patches. Continuing our example, if you patch _TrapA and then JSR to the original _TrapA code, you will have changed the contents of the stack. And believe it or not, some of the patches are involved enough that they look two or three stack levels high.

Hope this clears it up,

Andy Shebanow Mr. Clean, MacDTS Apple Computer, Inc.

●●●

From: parent@apple.com (Sean Parent)
Subject: Re: Tail patches

I have written lots of patches of all kinds in the past and I can tell you that they are not simple. If you are writing an application then there is no good reason to patch a trap that I have been able to determine... so don't. If you are writing an INIT, or some funky piece of psuedo-system software then you are walking a fine compatibility line to begin with and you should try to avoid patches.

Take a close look at your code to see if there is some way around the problem. For example: If you are writing an INIT and you wish to display an alert after the system has started up then don't patch SystemTask or GetNextEvent but use the notification manager. Or, if you just need some time then install a driver and get idle time that way.

If you decide you must patch a trap then try to avoid tail patches for the reasons mentioned previously. Patches to ROM routines do depend on return address (I know, I wrote one that does). With any kind of patch the rule is "take only picture, leave only footprints." Live by it. Restore EVERYTHING you can that you disturb. When chaining to the next routine restore ALL registers to their previous values, this can be done by reserving space on the stack at the start of your routine and moving the address that you will continue to into this space. At the end of your patch then restore all registers and do an RTS. The reason for this is that there are patches that upon checking the return address make assumptions about what is in the registers at that particular moment.

Be aware that:

- Multifinder keeps separate trap tables for each application.
- On 64K ROM machines the patch must be in the first 64K of RAM.
- If you remove your patch someone may have patched it after you so don't remove their patch also.
- Some traps dispatch to the same routine. For example _write async, and _write sync, dispatch to the same routine with the trap number in register D1 so you can determine which call was being made.
- Your patch may NEVER get called if something in the system (like Multifinder) replaces the routine entirely.

If you do write patches be prepared for your software to break even if you write it very carefully. Warn your employer and understand that you need a good update service in place to fix problems that may arise with the next release of system software. Build a maintenance clause into your contract if you are working as a contractor.

And before you write the patch, send DTS a link. They are pretty good at talking people off the roof. Give them a chance.

Sean Parent

•••

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: Menu font list appearing as font (source to MDEF)

In article <32498@ucbvax.BERKELEY.EDU> oster@dewey.soe.berkeley.edu.UUCP David Phillip Oster) writes:

> >To have a list of font names, each name in its own font (for example the
> >string "New York" in the New york font.) you need your own MDEF. It isn't
> >worth writing though, since there are at least two commercial products
> >that do this for you, uniformly, at INIT time. MenuFont 2, and Font/DA
> >Juggler, I think, are their names. It is much better to do this at INIT
> >time, since it is very slow to read in the font data, and create the
> >bitmaps. An INIT can pre-render all the fonts, and save them off to disk,
> >and only build new ones when there are changes.

This is a good idea; it can't be stressed enough that you should only do the menu rebuilding when there are changes in the available fonts,though. I'd hate to have to wait for an INIT to do this every time I restarted.

I think Claris has the right idea. All their programs with Font menus save them in a common data structure and a common file in the system folder, and each application comes with the necessary code to check the font list stored in that file against the current font list, rebuild the menu if necessary, and display and track the menu. It would be nice if someone would put code to do this in the public domain, or if Apple were to include it in system software. Probably the best way to do it would be to use the Resource Manager to store the data structures. The menu itself could be stored in a PICT, and the rectangles for each font name would be stored in an 'nrct'; a 'STR#' would hold the alphabetized list of font names.

Here's an old PICT MDEF I wrote. The extension to variable-sized elements should be pretty easy. Anyone else have useful code they can contribute to this cause?

```
#define ItemHeight 16

PASCAL void
mdef(message, menuid, r, p, which)
short message;
short **menuid;
Rect *r;
long p;
short *which;
{
	PicHandle pic = GetPicture(**menuid);
	short new;
	Rect invert;
	Point pt;

	switch (message) {
	case 0:
		/* draw the menu */
		DrawPicture(pic, r);
		break;
	case 1:
		/* choose and hilight */
		BlockMove((Ptr)&p, (Ptr)&pt, 4);
		if (PtInRect(pt, r)) {
			/* find the item */
			new = ((pt.v - r->top) / ItemHeight) + 1;
			if (*which != new) {
				if (*which != 0) {
					/* unhighlight which */
					SetRect(&invert, r->left,
```

```
                                        r->top + ((*which - 1) * ItemHeight),
                                        r->right, r->top + (*which * ItemHeight));
                                InvertRect(&invert);
                        }
                        /* highlight the item */
                        SetRect(&invert, r->left, r->top + ((new - 1) * ItemHeight),
                                r->right, r->top + (new * ItemHeight));
                        InvertRect(&invert);
                        *which = new;
                }
        }
        else if (*which != 0) {
                /* unhighlight which */
                SetRect(&invert, r->left, r->top + ((*which - 1) * ItemHeight),
                        r->right, r->top + (*which * ItemHeight));
                InvertRect(&invert);
                *which = 0;
        }
        break;
case 2:
        /* calculate size */
        (*menuid)[1] = (*pic)->picFrame.right - (*pic)->picFrame.left;
        (*menuid)[2] = (*pic)->picFrame.bottom - (*pic)->picFrame.top;
        break;
}
}
```

>If the INIT is also a
>control panel device, it can give the user a place to say that he wants to
>turn the mechanism off for unusual fonts that don't contain legible
>characters for their own name. Symbol is one such font, but any upper case
>only font with a mixed-case name would qualify.

Hmm. I don't know that this rule is correct. Might there not be symbol fonts that were not upper-case-only? And I confess that I'm mystified by what the mixed-case font name has to do with it.

Also, there are some Script Manager compatibility issues here that aren't clear to me. Perhaps some nice person at Claris could tell us how their applications deal with Symbol font and others of like kind, and how they deal with the international environment.

>I've written this feature into at least two prototype products that never
>saw the light of day. If you must do it, do it as an INIT, make it work
>everywhere, and compete with the other guys.

Yeah, well, I think the last thing we need is more INITs. Especially more overpriced single-function INITs. I'd rather have someone give me a code resource or two that I can just plug into my program and fly with.
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: mahboud@kinetics.com (Mahboud Zabetian,Kinetics,4511,9457194,)
Subject: Re: How can a CDEV "talk" to its INIT?

From article <8183@cs.yale.edu>, by kishon-amir@CS.Yale.EDU (amir kishon):

> I would like to write a CDEV that can change the global variables of a
> patch to GetNextEvent (the patch was done by an INIT that I wrote).
> How can I have both the CDEV and the patch (INIT) point to the same
> global variables, so the CDEV can change those values? Or are there
> any other ways to do that? namely, to keep some static global
> parameters referenced both by a CDEV code resource and an INIT code
> resource? I am working with Think C 4.0. HELPPPPPPPPPPPPPPPPPPPPP
>
> Thanks,
>

> -amir

Make the first instruction in your INIT be a JMP to the third instruction. Make the second instruction be a JMP to code that will set up your globals for you(these globals will probably be imbedded in your patch).

Now when you change the settings in the cdev, load in your INIT resource, and jump to 2(A0) where A0 is the beginning of the code.

Or if you can wait until reboot before changing things around, then just store that info in a resource and read it on the next reboot.

Mahboud Zabetian mahboud@kinetics.com

●●●

From: jholt@adobe.COM (Joe Holt)
Subject: Re: How do I install a driver with an INIT? (with Code)
Keywords: INITs driver

Here's a small code resource INIT I use to open a driver at startup time. I write the guts of my "INIT" as a driver, then paste this resource into it, and I'm off. With Think C, it even handles multi-segmented drivers. If you have any questions, just ask.

-------- CUT HERE --------

```
/**

  OPEN DRIVER INIT

  Think C 4.0

  This bit of code should be compiled as a Code Resource, Type INIT, ID 0 (or
  whatever you like).  Change the name DRIVER_NAME to the name of the driver
  you want opened at start up.  Place this compiled INIT resource in the
  driver and change the driver's file type to INIT.

  When this starts up, it finds an empty slot in the driver unit table and
  sticks your driver there.  It then opens your driver.  You should write the
  driver to handle whatever ref num it's assigned.

 **/


/**------------------------------------------------------------------------
 **
 **    Compilation Flags
 **
 **/

/**

  MULTI_SEGMENT generates code which is needed to INIT a multi-segmented
  driver.  Single-segment drivers can be inited also, so there is no logical
  requirement to ever turn this off, except where code size is a concern.

 **/

#define MULTI_SEGMENT    1


/**------------------------------------------------------------------------
 **
 **    Include Files
 **
 **/
```

```
    /* none */


/**----------------------------------------------------------------------
 **
 **    Private Macros
 **
 **/

/**

  UNIT_ENTRIES_NEEDED and FIRST_POSSIBLE_UNIT are used when determining the
  driver reference number for your driver.  This assigns the driver reference
  number dynamically, looking at FIRST_POSSIBLE_UNIT and on up for an unused
  number.  It increases the unit table size to UNIT_ENTRIES_NEEDED if
  there aren't that many unit numbers.  This occurs on Mac Pluses and Mac SEs
  with pre-System 6.0, where the unit table defaults to forty-eight entries,
  which doesn't leave any room for custom drivers like ours.

**/

#define UNIT_ENTRIES_NEEDED     (64)
#define FIRST_POSSIBLE_UNIT     (48)

#define DRIVER_NAME             "\p.your name here"


/**----------------------------------------------------------------------
 **
 **    Private Functions
 **
 **/

int main(void);

main()
{
    static char     driverName[] = DRIVER_NAME;
    int             theID;
    IOParam         ioPB;

    asm {
            movem.l D3-D4/A2-A4, -(A7)
            movea.l A0, A4


/**

  Change the resource attributes of our DRVR so that it will be loaded into
  the System Heap and locked.

 **/

            clr.b   -(A7)
            _SetResLoad

            clr.l   -(A7)
            move.l  #'DRVR', -(A7)
            pea     driverName
            _GetNamedResource
            move.l  (A7)+, D3
            beq     @abort

            move.l  D3, -(A7)
            move.w  #resSysHeap + resLocked, -(A7)
```

```
            _SetResAttrs
            move.l  D3, -(A7)     ; so that the res handle will be in SysHeap!
            _ReleaseResource

            move.b  #1, -(A7)
            _SetResLoad


/**

  Pick an unused driver reference num for our DRVR and renumber the resource
  to use this new reference number.  We first increase the number of entries
  in the unit table if less than UNIT_ENTRIES_NEEDED, then begin looking at
  FIRST_POSSIBLE_UNIT.

 **/

            clr.l   -(A7)
            move.l  #'DRVR', -(A7)
            pea     driverName
            _GetNamedResource
            move.l  (A7)+, D3
            beq     @abort

            move.l  D3, A3
            move.l  A3, -(A7)
            _DetachResource
            move.w  #UNIT_ENTRIES_NEEDED, D1
            sub.w   UnitNtryCnt, D1
            ble.s   @1

            bsr     @updrivers
            bne     @abort

@1:         movea.l UTableBase, A1
            move.w  #FIRST_POSSIBLE_UNIT * 4, D4
            move.w  #FIRST_POSSIBLE_UNIT, D3
@2:         cmp.w   UnitNtryCnt, D3
            bgt     @abort

            tst.l   0(A1, D4.w)
            beq.s   @3

            addq.w  #4, D4
            addq.w  #1, D3
            bra.s   @2

@3:         move.w  D3, D0
            not.w   D0
            movea.l (A3), A0
            dc.w    0xA43D              ; _DrvrInstall 4
            bne     @abort

            movea.l UTableBase, A1
            movea.l 0(A1, D4.w), A0
            _HLock
            movea.l (A0), A2
            move.l  A3, (A2)
            movea.l (A3), A3
            move.w  (A3), 4(A2)
            bset    #6, 5(A2)


/**

  Now set the resources which THINK C uses for drivers to load into the System
```

```
  Heap as locked, also.  We especially want to lock down any DCOD resources
  because the THINK C segment loader does a MoveHHi() on em when first loaded,
  which is not a good idea in the System Heap at INIT time.

 **/

            clr.b   -(A7)
            _SetResLoad

            clr.l   -(A7)
            move.l  #'DATA', -(A7)
            move.w  #1, -(A7)
            _Get1IndResource
            move.l  (A7)+, D3
            beq     @abort

            move.l  D3, -(A7)
            move.w  0x18(A2), D0
            not.w   D0
            lsl.w   #5, D0
            or.w    #0xC000, D0             ; DATA resource is always ID 0
            move.w  D0, -(A7)
            clr.l   -(A7)
            _SetResInfo
            move.l  D3, -(A7)
            move.w  #resSysHeap + resLocked, -(A7)
            _SetResAttrs
            move.l  D3, -(A7)
            _ReleaseResource


#if MULTI_SEGMENT

            subq.w  #2, A7
            move.l  #'DCOD', -(A7)
            _Count1Resources
            move.w  (A7)+, D4
            beq.s   @DCODout

DCODtop:    clr.l   -(A7)
            move.l  #'DCOD', -(A7)
            move.w  D4, -(A7)
            _Get1IndResource
            move.l  (A7)+, D3
            beq     @abort

            move.l  D3, -(A7)
            pea     theID
            clr.l   -(A7)
            clr.l   -(A7)
            _GetResInfo

            move.l  D3, -(A7)
            move.w  0x18(A2), D0
            not.w   D0
            lsl.w   #5, D0
            move.w  theID, D1
            and.w   #0xF01F, D1
            add.w   D1, D0
            move.w  D0, -(A7)
            clr.l   -(A7)
            _SetResInfo
            move.l  D3, -(A7)
            move.w  #resSysHeap + resLocked, -(A7)
            _SetResAttrs
            move.l  D3, -(A7)
```

```
            _ReleaseResource
            subq.w  #1, D4
            bne.s   @DCODtop

#endif    /* if MULTI_SEGMENT */

DCODout:    move.b  #1, -(A7)
            _SetResLoad

            bra.s   @open


;**
;
;    Increase the number of available entries in the unit table.
;
;**

@updrivers: move.w  UnitNtryCnt, D0
            add.w   D1, D0
            mulu    #4, D0
            _NewPtr SYS+CLEAR
            bne.s   @9

            move    SR, -(A7)
            move    #0x2600, SR
            movea.l A0, A1
            movea.l UTableBase, A0
            move.w  UnitNtryCnt, D0
            mulu    #4, D0
            _BlockMove
            _DisposPtr
            move.l  A1, UTableBase
            add.w   D1, UnitNtryCnt
            move    (A7)+, SR
            moveq   #0, D0
@9:         rts
    }
open:
    ioPB.ioNamePtr = (StringPtr)driverName;
    ioPB.ioPermssn = 0;
    PBOpen(&ioPB, FALSE);
abort:
    asm {
            movem.l (A7)+, D3-D4/A2-A4
    }
}
```

•••

From: t-alexc@microsoft.UUCP (Alex CHAFFEE)
Subject: Re: cdev - INIT Data Exchange
Keywords: cdev, INIT

In article <2695@cooper.cooper.EDU> joseph@cooper.cooper.EDU (Joe Giannuzzi) writes:

> I am writing a cdev that contains an INIT. I would like to
> exchange data between the two, but so far my attempts have
> failed. Does anyone have any recommendations about the best
> way to do this? Also, I have one DITL item that is an ICON.
> What is the best way for me to change the icon that it
> displays? Thanks.
>
> Joseph -> joseph@cooper.cooper.edu OR cmcl2!cooper!joseph

My favorite way is to have the INIT write a resource of type 'ADDR' into the preferences file (which should be buried safe in the system folder). The contents of this resource is just a longword pointing to the location of the shared data -- it can be a handle, or a pointer, or the proper value of A4. This is done once at startup, which is fine because the INIT code and data should be locked in memory for the duration..

This sounds kludgey, but it's actually quite clean and compatible. The INIT is guaranteed a writable system folder, even if it's been launched from a server. The existence and validity of the resource provides a quick way for the cdev to tell if the INIT's been loaded. The preferences file is going to keep the same name, even if the user renames the actual cdev/INIT file, so it'll be easy to find. Plus you don't have to walk through the system heap map, or install a driver, or any of the 1001 nasty alternative methods I've heard of...

As for your ICON, can't you just set the itemHandle to the new ICON resource (after GetResourceing it in from disk)? I thought that was how the Dialog Manager dealt with ICONs, but I'm not sure.

- Alex Chaffee
  chaffee@reed.bitnet or t-alexc@microsoft.uucp

●●●

From: lsr@Apple.COM (Larry Rosenstein)
Subject: Re: cdev - INIT Data Exchange

In article <1990Jun5.142604.11826@asterix.drev.dnd.ca> louis@asterix.drev.dnd.ca (Louis Demers) writes:

> What happens if you disable the INIT and it is not loaded. Your
> resource is still there pointing to some area it has no right to

I would place a magic series of bytes in memory and have the cdev check that the resource points to the expected magic bytes.

I've used this technique before, but users complained that the file in their system folder was modified each time they booted, so it was always being backed up.

Larry Rosenstein, Apple Computer, Inc.

●●●

From: cak3g@astsun7.astro.Virginia.EDU (Colin Klipsch)
Subject: CDEV/INIT Data Exchange, another way
Keywords: cdev, INIT, system heap

In article <8560@goofy.Apple.COM> lsr@Apple.COM (Larry Rosenstein) writes:

>In article <1990Jun5.142604.11826@asterix.drev.dnd.ca>
>louis@asterix.drev.dnd.ca (Louis Demers) writes:
>> What happens if you disable the INIT and it is not loaded. Your
>> resource is still there pointing to some area it has no right to
>
>I would place a magic series of bytes in memory and have the cdev check
>that the resource points to the expected magic bytes.
>
>I've used this technique before, but users complained that the file in
>their system folder was modified each time they booted, so it was always
>being backed up.

Here's my experience in writing TappyType, for what it's worth. . .

The first technique I used was the one stated above: the INIT, on startup would store a pointer to my system heap variables in a resource of type "TapT", ID#2 (if I remember correctly). This has the advantage of being more "compatible" than just about any other solution I've heard of, but which has the above disadvantage. Plus, if the user is especially mischievous and does things like: locks the CDEV file, temporarily moves it to another folder, deletes the resource, etc., you've got problems. Using an entire auxiliary file for CDEV-INIT communication is subject to the same problems, and more.

Of course, you can argue that anyone pretentious enough to commit any of these crimes against your CDEV deserves what he or she gets.

The next method I used -- briefly -- was searching the system heap for a block of the right size, and which began with a particular magic string. After rereading the Memory Manager chapter and various tech notes, however, I soon realized that this is a BAD idea. It relies completely on the format of memory blocks, which very very very probably will change in the future at arbitrary times. I suppose you could search the system heap for a magic string without paying attention to block boundaries, but this still seems rather iffy.

The current solution is a bit of a hack, but requires no auxiliary resources, nor installing a driver:

The INIT installs the code and variables in system heap memory from a resource (remember to use DetachResource!), and patches a few traps. Among them is _AddResource. For CDEV-INIT communication, I needed a trap which could be called in some way that my patch could uniquely identify as being a call from the CDEV.

My patch looks at every AddResource call. If the call is made with a resource of my particular type, my particular ID, if the name pointer points to four bytes of zeros, and if the return address is near a particular magic string, then my patch assumes the call was made by my CDEV. It then puts a pointer to my system heap variables in the four bytes of zeros pointed to by the name pointer, which my CDEV can pick up and use. The patch ends by jumping to the original AddResource address that the INIT found on startup (with NGetTrapAddress).

Note that adding a resource which already exists is harmless, so the resource I "add" from the CDEV is just my options resource, which my CDEV has previously loaded anyway. If TappyType has not been installed, then my patch won't be there, and the the four bytes of zeros will remain just that after the call. Otherwise, if it's non-zero, it's a pointer to my system heap variables.

To be even more anal-retentive, one could do the following:

- check that the handle being added is a resource, and that it's the right size
- check to see that the current resource file is of type "cdev" and creator "TapT" (using GetFileInfo)
- pass a pointer to four zero bytes as before, but after the four bytes is also a pointer to the magic string; check for it

In fact, I like that last one much better, now that I thought of it. I think I'll rewrite it that way for the next version. . .

Hope this helps. AddResource is undoubtedly not the only trap you could use; it's just the first feasible one I thought of. The time overhead is low, particularly if you write it in assembly (as one should probably).

I invite (constructive) criticism on this method. Anyone from Apple see anything wrong with this? (Anyone from anywhere, for that matter?)

Colin Klipsch

●●●

From: murat@farcomp.UUCP (Murat Konar)
Subject: Re: cdev - INIT Data Exchange
Keywords: cdev, INIT

In article <1990Jun5.142604.11826@asterix.drev.dnd.ca> louis@asterix.drev.dnd.ca (Louis Demers) writes:

> >t-alexc@microsoft.UUCP (Alex CHAFFEE) writes:
> >
> >>My favorite way is to have the INIT write a resource of type 'ADDR'
> >>into the preferences file (which should be buried safe in the system
> >>folder). The contents of this resource is just a longword pointing
> >>to the location of the shared data -- it can be a handle, or a
> >>pointer, or the proper value of A4. This is done once at startup,
> [...]
> > What happens if you disable the INIT and it is not loaded. Your
> > resource is still there pointing to some area it has no right to
> > claim. Upon shutdown, Do you remove this resource ? what happens
> [...]

I embed a 4 character string (OSType) into my patch code at a known offset from its address and check for it before writing there. Works really well.

--
Murat N. Konar

•••

From: urlichs@smurf.sub.org (Matthias Urlichs)
Subject: Re: CDEV/INIT Data Exchange, another way
Keywords: cdev, INIT, system heap

In comp.sys.mac.programmer, article <55103@microsoft.UUCP>, benw@microsoft.UUCP (Ben WALDMAN) writes:

> < On cdev-INIT communication:
>
> < The way I do it is to write my what's really my INIT as a driver.
> < Then, the actual INIT resource opens the driver by name, putting into the
> < system heap. (And the driver's open routine patches the traps I want to
> < patch, etc.). The driver also provides a status call, which returns the
> < address of its globals.
> <
> < The cdev, when it wants to communicate with the INIT, can simply
> < look through the unit table (this is described in a tech note), and find
> < the driver (by name). Then, the cdev can make a status call to the driver,

When you already know the driver's name, why not do an OpenDriver(name)? That'll get you its refnum much easier and safer (WRT compatibility). Apple specifically recommends not to scan the unit table if at all possible.

> < and, voila, the status call returns the info the driver needs. In my case,
> < I returned a pointer to the INITs globals (the INIT is locked in memory),
> < but you could, of course, return a handle, or an address of a function you
> < wanted to call, etc.

If you want the driver to do anything, it'd be much cleaner just to call the driver through _Control and/or _Status calls, no? That way the whole thing will also work if the user boots with version X of your driver, then installs Y (not necessarily greater than X), and opens the control panel...

> < The scheme fails if a dorky user changes the name of the DRVR resource
> < with ResEdit, but will still succeed if the DRVR gets renumbered.

You'll have to check in the unit table for a free refnum and install your driver there. You may have to make the unit table bigger; don't forget to zero the new table, record the new size in the appropriate global, don't free the old table because you don't know where it came from, and turn off interrupts while you do it.
--
Matthias Urlichs -- urlichs@smurf.sub.org -- urlichs@smurf.ira.uka.de

•••

From: jholt@adobe.COM (Joe Holt)
Subject: Informative INITs (code incl.; was: Re: Need help w/ Public Folder)

michael wrote:

> > ASIDE: It sure seems to me that INIT's need some way to communicate with
> > users other than just X-ing themselves out. I've been thinking about a
> > mechanism where if there is a problem the user can read about it in the
> > Chooser (or Control Panel) when they bring up the normal UI. The UI
> > would say something like, "Couldn't find a Public Folder" or "AppleTalk
> > is not turned on." What do people think of this idea?

I ran into the totally uninformative nature of x'ing the INIT icon out at boot time while writing Flash (an AppleTalk file transfer enhancement -- similar to Public Folder -- hello michael, nice to meet you!). I believe there are on the order of forty or fifty different reasons why Flash wouldn't install itself. Everything from "AppleTalk not installed" and "Yer

system's too old" to "out of memory" and "Flash is damaged". A lot of help an X is when joe user is trying to track down the problem.

Of course, bringing up an error window at INIT time is not only contrary to guidelines and tough to do but also just plain ugly.

I solved the problem by using Apple's Notification Manager.

If Flash has a problem, it gets a string from a STR# resource which describes the problem and puts it into a pointer in the system heap. It also loads a very small code resource and creates a Notification Manager structure. The code resource is a NM response procedure (read "completion routine") which gets rid of the structure and string and then itself.

Flash sets up the NM structure and posts a NM message. Flash then cleans up its act like a good boy scout and leaves no trace of itself. At this point the only things left around are the string, the code resource (detached) and the NM structure.

The NM message stays dormant until the Mac has finished booting and begins processing events (most likely in the Finder). Then the NM displays the message in an alert and the user gets a nice informative message. (If you have Flash and want to see this, take the file "ADSP" out of your system folder and reboot.)

When the user closes the note, the code stub gets control. It disposes of the string, the NM structure and finally itself, leaving no remains.

It works very well. It requires no disk I/O or temporary files, occupies about 90 bytes of system heap plus the length of your message, and has the advantage of being 100% compatible with current and future systems, without introducing a new concept to the user (e.g. "Go to the Chooser to read startup error messages" -- yuck!).

What follows are the three Think C files StartupError.h, StartupError.c, and StartupError RESP.c. The first two are used within your INIT's project. The third is the code resource which must be compiled separately. If there are any questions that might be of interest to the net, please post them here.

-------- CUT HERE FOR StartupError.h --------

```
/************************************************************************
 ***
 *** StartupError.h
 ***
 *** Informative error messages from INITs
 ***
 *** History:
 ***   jhh 18 jun 90 -- response to news posting
 ***
 ***/

#ifndef _H_STARTUP_ERROR
#define _H_STARTUP_ERROR


/************************************************************************
 **
 ** Public Functions
 **
 **/

void
StartupError(int errorNumber);


#endif  /* ifndef _H_STARTUP_ERROR */
```

-------- CUT HERE FOR StartupError.c --------

```
/**----------------------------------------------------------------------
 **
 ** Include Files
 **
 **/

#include "StartupError.h"


/**----------------------------------------------------------------------
 **
 ** Private Macros
 **
 **/

/**

    T_NMInstall and T_Unimplemented are Mac toolbox trap numbers used to
    test for the existence of the Notification Manager.

 **/

#define T_NMInstall         (0xA05E)
#define T_Unimplemented     (0xA89F)

/**

    STARTUP_ERROR_STR_ is the resource ID of the STR# containing the
    error message corresponding to the error number passed to
    StartupError().

    RESPONSE_RESP is the resource ID of the RESP code resource compiled
    separately and stuck in your INIT's resources.

 **/

#define STARTUP_ERROR_STR_  (128)
#define RESPONSE_RESP       (128)


/**********************************************************************
 ***
 *** void    StartupError(int errorNumber);
 ***
 *** When your INIT runs into a problem, clean things up, show the X'ed
 *** version of your icon and call StartupError() with an error number
 *** corresponding to the message you want displayed.
 ***
 *** History:
 ***   jhh 18 jun 90 -- response to news posting
 ***
 ***/

void
StartupError(int errorNumber)
{
    register NMRec  *note;
    register Handle responseCode;
    register long   size;
    register THz    svZone;
    Str255          errorText;

/**

    Make sure we've got a Notification Manager.
```

```
 **/

    if (NGetTrapAddress(T_NMInstall, OSTrap) !=
            NGetTrapAddress(T_Unimplemented, ToolTrap)) {

/**

    All of the memory we allocate from here on out is in the System
    Heap.  First create a Notification Manager record and fill it in.
    Note that you can expand this notification method with sounds and
    icons by adding the appropriate code.  See the Notification
    Manager technote #184 for details.

 **/

        svZone = TheZone;
        TheZone = SysZone;
        note = (NMRec *)NewPtr(sizeof(NMRec));
        if (!note)
            goto exit;
        note->qType = nmType;
        note->nmMark = 0;
        note->nmSIcon = 0L;
        note->nmSound = (Handle)-1;

/**

    Get the error message corresponding to the error number given.
    For maximum performance, the STR# resource should be tagged
    "Preload" and not "System Heap".  This way, you can be sure
    the messages will be there even if memory space is the cause of
    the error.

    We create a pointer in the System Heap just big enough for the
    string and copy the string into it.  Point the NM record at this
    string.

 **/

        GetIndString(errorText, STARTUP_ERROR_STR_, errorNumber);
        size = *(unsigned char *)errorText;
        note->nmStr = (StringPtr)NewPtr(size);
        if (!note->nmStr) {
            DisposPtr(note);
            goto exit;
        }
        BlockMove(errorText, note->nmStr, size);

/**

    The response procedure also must be in a pointer in the System
    Heap.  You need to include the compiled code resource in your
    INIT's resources of type 'RESP'.

    Create a pointer just big enough for it and point the NM record
    at it, also.

 **/

        responseCode = GetResource('RESP', RESPONSE_RESP);
        if (!responseCode) {
            DisposPtr(note->nmStr);
            DisposPtr(note);
            goto exit;
        }
```

```
        size = GetHandleSize(responseCode);
        note->nmResp = (ProcPtr)NewPtr(size);
        if (!note->nmResp) {
            DisposPtr(note->nmStr);
            DisposPtr(note);
            goto exit;
        }
        BlockMove(*responseCode, note->nmResp, size);

/**

    Now post the note.  As soon as startup is complete, the NM
    will display the note for the user's edification.  Hurrah.

 **/

        NMInstall(note);
exit:
        TheZone = svZone;
    }
}
```

-------- CUT HERE FOR StartupError RESP.c --------

```
/**

    Compile this code in a separate project.  Set the project type to
    Code Resource, type 'RESP' ID 128.  Once compiled, you can build it
    and merge it into your INIT's resource file ("...project.rsrc") or
    build it into a separate file and use ResEdit to copy it in.  This
    code is independent of the INIT, so it can be used as-is for any
    INIT you write.

 **/


/**

    ToolScratch is an 8-byte area of low memory used by the Mac toolbox
    and other people as a temporary holding place.

 **/

extern long     ToolScratch : 0x09CE;


/*****************************************************************************
 ***
 *** pascal void    main(QElemPtr nmReqPtr);
 ***
 *** This is the code resource type 'RESP' which you need to compile
 *** separately and include among your INIT's resources.
 ***
 *** This code is called when the user closes the Notification Manager's
 *** note dialog.  The NM passes to us the address of the NM record which
 *** was set up by StartupError().  We use this to clean up and leave.
 ***
 *** This is written is assembly for size.  If you have problems with
 *** assembly, I s'pose it could be written in C, but the trick at the
 *** end would be hard to duplicate...
 ***
 *** History:
 ***   jhh 18 jun 90 -- response to news posting
 ***
```

```
 ***/

pascal void
main(QElemPtr nmReqPtr)
{
    asm {

/**

    First remove the note from the Notification Manager's notification
    queue.

 **/

            move.l      nmReqPtr, A0
            _NMRemove

/**

    Grab the string's address from the NM rec and dispose of it.  Then
    get rid of the NM rec itself.

 **/

            move.l      nmReqPtr, A0
            move.l      OFFSET(NMRec,nmStr)(A0), A0
            _DisposPtr
            move.l      nmReqPtr, A0
            _DisposPtr

/**

    Now the tricky part.  This code lives within a small block in the
    System Heap which we want to get rid of.  But it's not safe to
    dispose of the very block you're calling from!  So, we create a
    little bit of code in ToolScratch which does the dispose for us
    and execute it last.

 **/

            move.l      4(A7), A1
            move.l      #ToolScratch, A0
            move.l      A0, 4(A7)
            move.l      #0x2040A01F, (A0)   ; movea.l D0, A0 / _DisposPtr
            move.w      #0x4ED1, 4(A0)      ; jmp (A1)
            lea         main, A0
            move.l      A0, D0              ; Pascal clobbers A0 on exit
    }
}
```

-------- END --------

[I apologize for all of the spaces in the posting; I used them instead of tabs because of the funky tabs my terminal emulator gives.]

●●●

From: leipold@eplrx7.uucp (Walt Leipold)
Subject: Re: 24-Hour FKEY?

For anybody who's interested, the following is an FKEY (resource number 6) to toggle the Mac's time display from 12- to 24-hour mode. Install it with ResEdit or FKEY Manager. Have fun...

Disclaimer: I've only tested this FKEY with the American version of the System file.

{ This is a THINK Pascal source file. Compile it as a code resource of type FKEY (resource number [whatever you want], Custom Header option,Purgeable), and link it with Interface.lib & DRVRRuntime.lib }

```
unit FKEY1224;

    interface
        procedure main;

    implementation

        procedure main;
        var
            h: Handle;
            i: Intl0Hndl;
        begin
            h := GetResource('itl0', 0);
            if h = nil then
                SysBeep(10)
            else begin
                i := Intl0Hndl(h);
                if i^^.timeCycle = 0 then begin
                    i^^.timeCycle := 255;
                    ChangedResource(h);
                    UpdateResFile(0);
                    end
                else if i^^.timeCycle = 255 then begin
                    i^^.timeCycle := 0;
                    ChangedResource(h);
                    UpdateResFile(0);
                    end
                else
                    SysBeep(10);
                end;
        end;

end.
```

Walt Leipold

●●●

From: jpab+@andrew.cmu.edu (Josh N. Pritikin)
Subject: Re: VBL tasks Think C 4.0

> >I am writing an INIT that patches the JCrsrTask routine with my
> >own routine. I am confused about Think's SetUpA4 macro and the
> >IM SetUpA5 macro. They both seem to give me access to the globals

In general, SetUpA5 is used in applications and SetUpA4 is used in code resources. Check the manual for specifics. Since you writing an INIT, your globals will be accessed through A4. Read the part in the manual about INITs now, then read the rest of this. I will describe one of many ways to do what you want to do.

On entry, at bootup, D0 contains a pointer to the INIT resource. You should called RememberD0 and SetUpA4 (in addition to movem all the registers your going to change onto the stack). This set of calls will store the ptr in a local global variable (read, "not A4 referenced but it is static") and load A4 with it. If you want your INIT to stick around, you should RecoverHandle on A4, then DetachResource, HLock and HNoPurge (to be safe).

When your INIT is called again from JCrsrTask or the like, you need to SetUpA4, but DON'T RememberD0. The local global variable already has the correct value (the handle is locked down). You can exit normally using RestoreA4, etc. To be safe, you shouldn't change any registers as a side effect of your routine unless you know what your doing.

Always make sure the stack pointer doesn't get screwed up and happy debugging...

/* Josh Pritikin

•••

Chapter 3 Communications & Networking

Zones (how does one determine what zone one is in)
(with code) 62
MacTCP programming with THINK C 63
Serial driver - Why is CTSHold being set? 63
Problem with Apple's distributed MacTCP dnr.c routine
(bug has been fixed) 65

From: rmh@apple.com (Rick Holzgrafe)
Subject: Re: Zones (how does one determine what zone one is in with code)

In article <sZhJIpi00WBL03UUlC@andrew.cmu.edu> nf0i+@andrew.cmu.edu (Norman William Franke, III) writes:

> > I found out how to read the zones from some sample code from Apple, but
> > how does one determine what zone one is in? I thought it was the first
> > zone returned, but it just "broke". Does anyone know how to do this?

Yes. The Inside Mac documentation is sketchy and the MPW include files are incomplete, but it can be done and here's how:

```
GetMyZone (char buf[578])
{
    OSErr err;
    short atpRefNum, mppRefNum;
    short myNode, myNet;
    SendReqparms atp;
    BDSElement bds;


    err = OpenDriver ("\P.MPP", &mppRefNum);
    if (err != noErr) /* oops */

    GetNodeAddress (&myNode, &myNet);

    err = OpenDriver ("\P.ATP", &atpRefNum);
    if (err != noErr) /* oops */

    bds.buffSize = sizeof (buf);
    bds.buffPtr = buf;

    atp.ioCompletion = NULL;
    atp.ioRefNum = atpRefNum;
    atp.csCode = 255; // sendRequest

    atp.userData = 0x07000000;// GetMyZone
    atp.atpFlags = 0;
    atp.addrBlock.aNet = myNet;
    atp.addrBlock.aSocket = 6;
    // GetBridgeAddress is new, if not imp. use low-mem global
    // ABusVars.sysABridge
    // If bridge == 0 or net == 0, no bridge and no zones!
    atp.addrBlock.aNode = GetBridgeAddress ();
    atp.reqLength = 0;
    atp.reqPointer = (Ptr)0;
    atp.bdsPointer = (Ptr)&bds;
    atp.filler = 1;// NumOfBuffs -- honest!
    atp.timeOutVal = 1;
    atp.retryCount = 3;
    atp.atpFlags = 0;
    atp.retryCount = 3;

    PBControl ((ParmBlkPtr)&atp, false);
    err = atp.ioResult;

    if (err != noErr) /* oops */

    if (buf[0] > 32) // "Can't happen", it says here
        buf[0] = 32;
    /* buf now contains the name of your zone! */
}
```

The above won't compile as is (you'll need error handling at least), and I may have screwed something up by excerpting this from a larger routine, but it *does* come from a working routine. It's in MPW 3.0 C. Hope i t helps.

-
Rick Holzgrafe

●●●

From: resnick@lees.cogsci.uiuc.edu (Pete Resnick)
Subject: MacTCP programming with THINK C
Keywords: think ,mactcp

I have gotten enough request for the changes I have made to the MacTCP header files that I think I will post them here. There are only a few changes that I have made so far, but there may be others that I haven't used yet.

In MacTCPCommonTypes.h, change the "ifndef/include" definition to look like this:

```
#ifndef _MacTypes_
#include <MacTypes.h>
#endif /* _MacTypes_ */
```

In UDPPB.h, comment out the word "PASCAL" in the declaration of UDPNotifyProc so that it looks like this (THINK C doesn't like a typedefed PASCAL function):

```
typedef /*pascal*/ void (*UDPNotifyProc) (
       StreamPtr udpStream,
       unsigned short eventCode,
       Ptr userDataPtr,
       struct ICMPReport *icmpMsg);
```

Do the same thing in TCPPB.h for TCPNotifyProc:

```
typedef /*pascal*/ void (*TCPNotifyProc) (
       StreamPtr tcpStream,
       unsigned short eventCode,
       Ptr userDataPtr,
       unsigned short terminReason,
       struct ICMPReport *icmpMsg);
```

You will not be able to use these typedefs in your own programs; they are only to deal with other declarations in your header files. In your code, declare your notify procs as 'PASCAL void whatever' with all of the parameters as they are declare in the typedef.

That's all I have found so far, except that in my copy of TCPPB.h, instead of the csCode for receive buffer return being TCPBfrReturn as documented, it was TCPRcvBfrReturn. I changed it to TCPBfrReturn.

pr
--
Pete Resnick (...so what is a mojo, and why would one be rising?)

●●●

From: chesley@goofy.apple.com (Harry Chesley)
Subject: Re: Serial driver - Why is CTSHold being set?

In article <RANG.90Jun5232240@derby.cs.wisc.edu> rang@cs.wisc.edu (Anton Rang) writes:

> What I do:
>
> 1. Call RAMSDOpen (on port A).

According to a recent tech note, it's now better to do just an OpenDriver, rather than RAMSDOpen. RAMSDOpen was needed to load the RAM driver in the original Macs, but now that driver is standard. I don't believe this could cause any problems at this point, but OpenDriver is better for the future.

> 2. Call SerReset for the port A input and output drivers.

Triple check the parameters to this call. It's quite possible that something is off, and so you're sending at the wrong baud-rate, number of data bits, or something.

> Calling SerStatus reports that 'holdCTS' is on. Presumably, this
> means that the driver is convinced my modem isn't ready. But (1) my
> modem has CTS forced on, and (2) all the communications software I
> have works fine.

CTS is used for signal flow control. However, I'm quite sure the driver doesn't listen to it unless you explicitly turn it on via SerHShake, so the problem should be earlier than that. If you are trying to talk to a modem, it's possible that the modem isn't listening until some modem signal is asserted (like DTR), though other comm software working suggests that's not it. Don't, however, count on the modem wires going through from the modem to the Mac, as there are as many different cables out there as there are permutations of the wires in the cables (unless you've buzzed your cable out yourself).

> P.S. I'd like to wind up with an application which can handle
> incoming data while in the background under MultiFinder.
> Anyone have any suggestions on the best (or worst) ways
> to go about this? I've thought about setting up a big
> input buffer and periodically checking how many characters
> are in the buffer, using PBRead to extract them, and either
> queueing them for later or processing them. Will this work?

That's the way I'd do it. You can also set up an asynchronous read.

•••

From: jamesbo@microsoft.UUCP (James Borquist)
Subject: The prefered AppleTalk interface, is it from hell or what?

Hi there! I'm trying to use the perfered interface to transmit some information between two computers via ATP. I have been doing this with the alternative interface for quite some time now with the best of success, but for some reason, I just can't seem to get this to work with the prefered interface. I have two programs. They are quite simple tests, they don't do much, and they don't work. Could someone look at the following code fragments and tell me what I am doing wrong before I go completely crazy?

This is from the server:

```
with requestPB { ATPParamBlock } do
  begin
    ioCompletion := nil;
    atpSocket := mySocket; { byte }
    reqLength := 578;
    reqPointer := @requestBuffer; { array[1..578] of byte }
  end;
result := PGetRequest(@requestPB, true);
while 1<>2 do
  if requestPB.ioResult <= 0 then
    begin
      with responsePB { ATPParamBlock } do
        begin
          atpSocket := mySocket;
          atpFlags := atpEOMvalue;
          addrBlock := requestPB.addrBlock;
          bdsPointer := @responseBDS; { BDSType }
          filler0 := 1;
          bdsSize := 1;
          transID := requestPB.reqTID;
        end;
      result := PSendResponse(@responsePB, false);
      with requestPB do
        begin
          ioCompletion := nil;
          atpSocket := mySocket;
          reqLength := 578;
```

```
            reqPointer := @requestBuffer;
          end;
        result := PGetRequest(@requestPB, true);
      end;
```

And this is from the Cient:

```
    with requestPB do
      begin
        atpFlags := atpXOvalue;
        addrBlock := address; { from PLookupName and NBPExtract }
        reqLength := 578;
        reqPointer := @requestBuffer; { array[1..578] of byte }
        bdsPointer := @responseBDS; { BDSType }
        numOfBuffs := 1;
        timeOutVal := 8;
        retryCount := 3;
      end;
    result := PSendRequest(@requestPB, false);
```

Please, somebody help me!!!

Thanks,
James Borquist (uunet!uw-beaver!microsoft!jamesbo)

●●●

From: zben@umd5.umd.edu (Ben Cranston)
Subject: Problem with Apple's distributed MacTCP dnr.c routine (bug has been fixed)
Keywords: MacTCP Domain Volume

There is a problem with the version of dnr.c distributed by Apple with early version of MacTCP. It causes applications not to be able to initialize the Domain Name Resolver when running from a volume that is not the system volume (the one containing the current system file).

In routine OpenOurRF() the code does a PBHGetFInfo scan of the system folder looking for a file with type 'cdev' and creator 'mtcp'. When it finds one it calls OpenResFile on the name returned by the GetFInfo and returns the status to its caller, OpenResolver. Due to the vagaries of configuration on 512e machines (:-) the mainline ignores the status.

The name returned by the GetFInfo scan is a "bare" name, e.g., "MacTCP". It is not a pathname like "Harddisk:System Folder:MacTCP". So, the OpenResFile will search the directory containing the executing application and typically not find MacTCP, but then the Poor Man's Search Path will cause the search of the System Folder and MacTCP will be found. The code in OpenResolver will then proceed to read the 'dnrp' resource into memory and call it.

Those who know the PMSP will now understand the problem: the PMSP does NOT cross volume boundaries! So, if you put your program on a floppy it suddenly stops working.

After finding this out I took a look at the NameToAddress Hypercard XCMD and found they had modified the code to solve this problem. What they did, and what I have done, is to do a SetVol to the system folder before the OpenResFile so it will search the System Folder rather than the folder containing the application.

But MY code saves and restores the old SetVol and theirs does not :-).

```
/* ! */ marks new lines:

/* OpenOurRF is called to open the MacTCP driver resources */
short OpenOurRF()
{
    SysEnvRec info;
    HParamBlockRec fi;
    Str255 filename;
#ifdef ZBEN                 /* ! */
    short savevol;                  /* ! */
```

```
    OSErr status;            /* ! */
#endif                   /* ! */

    SysEnvirons(1, &info);
    fi.fileParam.ioCompletion = nil;
    fi.fileParam.ioNamePtr = &filename;
    fi.fileParam.ioVRefNum = info.sysVRefNum;
    fi.fileParam.ioDirID = 0;
    fi.fileParam.ioFDirIndex = 1;
    while (PBHGetFInfo(&fi, false) == noErr) {
/* scan system folder for driver resource files of specific type & creator */
    if (fi.fileParam.ioFlFndrInfo.fdType == 'cdev' &&
        fi.fileParam.ioFlFndrInfo.fdCreator == 'mtcp') {
/* found the MacTCP driver file */
#ifdef ZBEN                  /* ! */
        GetVol(nil,&savevol);               /* ! */
        SetVol(nil,info.sysVRefNum);        /* ! */
        status = OpenResFile(&filename);    /* ! */
        SetVol(nil,savevol);         /* ! */
        return(status);              /* ! */
#else                        /* ! */
#ifdef MPW3.0
        return(OpenResFile(&filename));
#else
        return(OPENRESFILE(&filename));
#endif
#endif                       /* ! */
    }
/* check next file in system folder */
    fi.fileParam.ioFDirIndex++;
    fi.fileParam.ioDirID = 0;
    }
    return(-1);
}
```

•••

Chapter 4 Compilers & Development Environments

Ramblings about MacApp 68
To all users of ThinkC and MultiFinder... 69
MPW Pascal and self referencing 70
CClusters in Think C 71
LSP 3.0 bug (?) 71
global data in Think C 72
THINK Pascal interfaces (was SysEnvirons) 72
Alternative pkg interfaces for Pascal 73
Bug in THINK Pascal? My bug? 77
A possible bug in THINK C 4.0.1 77
32k arrays (in Think Pascal) 78
Multiple Inheritance for HandleObjects in C++ 79

From: keith@Apple.COM (Keith Rollin)
Subject: Re: Ramblings about MacApp

In article <1989Nov2.190025.14568@polyslo.CalPoly.EDU> tdrinkar@cosmos.acs.calpoly.edu.UUCP (Terrell Drinkard) writes:

> >New subject:
> > Can someone out there give me some idea as to the usefullness of
> >MacApp? I'm not even real sure as to what it is, actually. But it
> >was noted in my MPW docs and I am interested.

Let me put it this way: I won't program without MacApp anymore. And I know that anyone else in DTS who had mastered MacApp and needs to write a real program feels the same way (unless your name is Paul, in which case you'll use ACL).

MacApp is an object oriented application framework. It is written in Apple's Object Pascal, but can also be accessed by C++. It gives you the basic frame- work of an application, and allows you to fill in the missing parts. For instance, all you need to do to get a working program that supported a full set of menus, multiple windows, window management, about box, scrollbars,memory management, exception handing, multifinder awareness, etc. is write about 50 lines of source code. Here is a little Nothing program that comes with MacApp that gives you all that I mentioned. Basically, all it does is create an Application object responsible for running your program (from the main event loop on down), and defines a procedure to be called when the window needs to be updated (TDefaultView.Draw):

```
program UNothing;

 uses
  UMacApp, UPrinting, Fonts;

 const
  kSignature = 'SS01';
  kFileType = 'SF01';

 type
  TNothingApplication = object(TApplication)
    procedure TNothingApplication.INothingApplication (itsMainFileType: OSType);
   end;

  TDefaultView = object(TView)
    procedure TDefaultView.Draw (area: Rect);
    OVERRIDE;
   end;

 var
  gNothingApplication: TNothingApplication;

 procedure TNothingApplication.INothingApplication (itsMainFileType: OSType);

 begin
  IApplication(itsMainFileType);
  RegisterStdType('TDefaultView', 'dflt');
  if gDeadStripSuppression then
   if Member(TObject(nil), TDefaultView) then
    ;
 end;

 procedure TDefaultView.Draw (area: Rect);
  OVERRIDE;

  var
   itsQDExtent: Rect;

 begin
  PenNormal;
  PenSize(10, 10);
```

```
  PenPat(dkGray);
  GetQDExtent(itsQDExtent);
  FrameRect(itsQDExtent);
  TextFont(ApplFont);
  TextSize(72);
  MoveTo(45, 90);
  DrawString('MacApp(');
  PenNormal;
 end;

begin

 InitToolBox;
 if ValidateConfiguration(gConfiguration) then
  begin
   InitUMacApp(8);
   InitUPrinting;
   New(gNothingApplication);
   FailNIL(gNothingApplication);
   gNothingApplication.INothingApplication(kFileType);
   gNothingApplication.Run;
  end
 else
  StdAlert(phUnsupportedConfiguration);
end.
```

In addition, MacApp gives you excellent development tools and debugging facilities, including an object inspector, a class browser, a view layout editor, discipline, writeln window, high-level breaks and tracing, performance tools, etc.

> > And if you don't mind, could someone also explain the benefits
> >of belonging to APDA?

You can buy MacApp...

Keith Rollin --- Apple Computer, Inc. --- Developer Technical Support

●●●

From: u-atgoat%ug.utah.edu@cs.utah.edu (Alan T Goates)
Subject: To all users of ThinkC and MultiFinder...
Keywords: Think C, MultiFinder, ResEdit

Don't you hate it when you have Think C already running, and you try to double click a project from the Finder. No dice, MultiFinder gives you an error message. Well, I got to poking aroung a bit and fixed this problem.
Here's how:

First, turn MultiFinder off and re-boot (if MF is on).
Then run ResEdit and open up MultiFinder.
Open up the 'mst#' resources.
Create a new one, and give it ID #102
Open it up as a 'STR#' type.
Add the two strings "File" and "Project".
Open up 'mst#' ID 101 as a 'STR#'.
Add the string "Open Project..."
    make sure the the ... is just one character (option-semi-colon).
Now go into the 'mstr' type and delete ID #102.
Save and re-boot with MultiFinder on.

That's it. Have Fun.

AL
●●●

From: keith@Apple.COM (Keith Rollin)
Subject: Re: MPW Pascal and self referencing

In article <939@uvicctr.UVic.CA.UUCP> franklin@uvicctr.UUCP (Katherine Franklin) writes:

> >I am writing an MPW Pascal tool, and I have written an invariant check
> >function. What I would like to do is call is this function many times
> >throughout the program and each time the invariant *doesn't* hold is to
> >write a message to the console.
> >
> >Unix C has a macro variable that allows you to pass the current line number
> >in the source code on to the object code to report during execution. I would
> >like to do this, or some other dynamic way.
> >
> >Is this possible ? if so, how ?
> >
> > Thanks,
> > Katherine

Katherine,

I don't know if there is a way to get the actual line number an instruction came from in MPW Pascal. However, there is a kludge that will give you the procedure name as long as you have compiled your program with Debugger names turned on. MacApp uses the following routine in its own debugger to determine where errors occured, or for monitoring tracing:

```
PROCEDURE GetProcName(ppc: Longint;VAR className, procName: MAName);
{ GetProcName returns the name of the procedure or function in
which ppc points.  If it is in a method, then it return's
the name of the method's class in className. }

   VAR
      pc, nextPC, limit:   Ptr;
      index:            INTEGER;

   BEGIN
   pc := Handle(ppc)^;
   IF (ord(pc) <> 0) & NOT Odd(ord(pc)) THEN
      BEGIN
      limit := Ptr(ord(pc) + 32767);
      WHILE (endOfModule(pc, limit, @procName, nextPC) = NIL) DO
         BEGIN
         IF ord(pc) >= ord(limit) THEN
            BEGIN
            className := '';
            procName := '';
            LEAVE;
            END
         ELSE
            pc := Ptr(ord(pc) + 2);
         END;

      index := pos('.', procName);
      IF index <> 0 THEN
         BEGIN
         className := copy(procName, 1, index - 1);
         END
      ELSE
         className := '';
      END
   ELSE
      BEGIN
      className := '';
      procName := '';
      END;
```

```
   END;
```

("endofmodule" is in the DisAsm library that comes with MPW.) From your assertion routine, call this routine with a pointer to the return address of the routine that called your assertion. You can get this return address with something like the following expression:

```
    Ord4(GetCurStackFramePtr) + 4
```

Where "GetCurStackFramPtr" is an inline procedure that looks like:

```
FUNCTION GetCurStackFramePtr: Ptr;
      INLINE $2E8E;              { MOVE.L A6,(A7) }
```

Hope this helps,

Keith Rollin --- Apple Computer, Inc. --- Developer Technical Support

●●●

From: lim@iris.ucdavis.edu (Lloyd Lim)
Subject: Re: CClusters in Think C

In article <468@adimail.UUCP> tel@adimail.UUCP (Terry Monks) writes:

> >[last time I asked such a question here I got only 3 responses, of which only
> >1 was of any value - which makes me think that not many people are actually
> >using Think C objects...]
> >
> >If I have a CCluster of objects, all of which have a PrintMe method, how do
> >I get the whole group to PrintMe? If I use DoForEach, it seems to me that I
> >need an intermediate routine that does nothing but call PrintMe for each
> >object passed, and that can't be efficent, can it?

Yes, the way it is currently set up you have to have a short little static routine to pass to DoForEach. I commented on how this seemed inelegant in a previous thread and Tom Lippincott replied with a cleaner way. After thinking about it a little bit, both methods are about the same in efficiency but Tom's suggestion doesn't need those static routines everywhere. Go back and look for his reply under the subject Re: OOP & algorithms if you areinterested.

Lloyd Lim Internet: lim@iris.ucdavis.edu (128.120.57.20)

●●●

From: hal@krishna.cs.cornell.edu (Hal Perkins)
Subject: Re: LSP 3.0 bug (?)

In article <XCFKSY@xavier.swarthmore.edu> Nick Jackiw writes:

> >Here's an annoying discrepancy I turned up in THINK Pascal (v3.0), which
> >can cause run-time behavior in the environment to differ from behavior
> >in a compiled application.

[sample code whose behavior changes]

> >   with aPtr^ do
> >   aPtr:=Ptr(ord(aPtr)+x+y+z)
> >
> >where x, y, and z are fields of whatever aPtr pointed to, they may or
> >may not be valid when the compiler sums the r-expr.
> >
> >I'm not sure I'm correct in saying this is a *bug*--I don't know what
> >the Pascal definition says in this case--but if it's not a bug in
> >pascal, it might be a bug in your code, so watch out.

The only bug here is that whoever wrote this code doesn't know Pascal well enough. Standard Pascal says that in the statement

```
with e do s
```

the statement s may not modify any of the variables referenced in the expression e. This is precisely to permit the compiler to optimize the statement by keeping the pointer expression in a register or whatever else is helpful.

Hal Perkins hal@cs.cornell.edu

●●●

From: ech@cbnewsk.att.com (ned.horvath)
Subject: Re: global data in Think C

From article <waIrjuG00UgyAPK458@andrew.cmu.edu>, by cc4b+@andrew.cmu.edu (Christopher Brian Cox):

> Rich,
>
> Face it, the 32k global data limit is a BUG!
> I have a little program here called p2c. It had 90+k of global data
> when I first compiled it. After removing the un-initialized arrays
> and allocating them at runtime, it had 48+k of global data.
> That's with Seperate STRS.
> The compiler should automatically allocate un-initialized arrays at
> runtime. There still shouldn't be a 32k limit.
>
> Chris Cox

Ease off. 32K is a restriction, an irritation at worst. It is routed in the fact that the MC68000 uses 16-bit signed offsets for address-register based addressing, and furthered by Apple's decision to use a program model that includes "globals are accesssed as negative offsets from a5."

There are ways around this for the compiler writer, but because of the inherent limitations of the processor chip, they all require explicit address arithmetic in the generated machine code. That means larger, slower programs.

You have a couple of alternatives. One you've identified: using NewPtr or NewHandle at run time for uninitialized arrays cut your "globals" requirement in half. That has the rather nice side effect that you can resize such arrays to match the problem size.

For large initialized arrays, consider "compiling" them into resources, or reading them in from your data fork. I won't argue that this is convenient, and you might justifiably claim that Motorola, Apple, and Think/Symantec have made your life a little tougher. The only compiler I know of that lets you duck the limit (for a price in speed and size) is Aztec C.

=Ned Horvath=

●●●

From: gft_robert@gsbacd.uchicago.edu
Subject: THINK Pascal interfaces (was SysEnvirons)

In article <1043@gargoyle.uchicago.edu>, dawyd@gargoyle.uchicago.edu (David Walton) writes...

>In article <41411@apple.Apple.COM> anderson@Apple.COM (Clark Anderson) writes:
[...]
>>LSP doesn't know what a SysEnvPtr is.
>
>Use a SysEnvRec, not a SysEnvPtr. IM Vol V is incorrect in its
>declaration, at least as far as I can tell. Check out Tech Note 129.

I had the same problem and came up with the same 'solution'.

While this is just an error in IM, it would be nice if THINK Pascal included proper interface files for the stuff that's defined internally in THINK Pascal. I remember when I ran across this problem I went to look for the interface

definition of the SysEnvirons() call. No dice. When THINK P. defines something internally, they just have a blank file for the interface. Why not put the proper interfaces in, so that we can see exactly what THINK P. wants, and comment them out, so that it doesn't interfere with whatever mechanism is currently in place?

Robert

= gft_robert@gsbacd.uchicago.edu

•••

From: ccc_ldo@waikato.ac.nz (Lawrence D'Oliveiro, Waikato University)
Subject: Alternative pkg interfaces for Pascal

Ever found the standard Pascal definition for SFGetFile/SFPGetFile to be a nuisance? I have. An SFTypeList of 4 elements is almost never what I want. Often I just want one type. Several times I've wanted a dynamic list of types, which could be of any length.

Also, call me irresponsible, but I think it's inelegant to have an assembly- language glue routine when a short inline will do the trick.

So here my alternative declarations of some toolbox calls. First of all, the disk-initialisation package in DiskInit.p:

```
Procedure DILoad;

   Inline
  $3F3C, $0002,       {move.w #2, -(sp)}
  $A9E9;       {_Pack2}

Procedure DIUnload;

   Inline
  $3F3C, $0004,       {move.w #4, -(sp)}
  $A9E9;       {_Pack2}

Function DIBadMount
  (
   where : Point;
   evtMessage : LongInt
  ) : Integer;

   Inline
  $4267,       {clr.w -(sp)}
  $A9E9;       {_Pack2}

Function DIFormat
  (
   drvNum : Integer
  ) : OSErr;

   Inline
  $3F3C, $0006,       {move.w #6, -(sp)}
  $A9E9;       {_Pack2}

Function DIVerify
  (
   drvNum : Integer
  ) : OSErr;

   Inline
  $3F3C, $0008,       {move.w #8, -(sp)}
  $A9E9;       {_Pack2}

Function DIZero
  (
   drvNum : Integer;
```

```
    volName : Str255
   ) : OSErr;

    Inline
   $3F3C, $000A,       {move.w #10, -(sp)}
   $A9E9;       {_Pack2}
```

And here are new definitions for most of the routines in Packages.p (IUCompString and IUEqualString are a little more complicated, so I leave them as glue). SFGetFile and SFPGetFile have their typeList arguments declared as "Ptr", so you can pass just about anything you like.

Oh, and StringToNum is a function, which I've found to be far more convenient, particularly when I only want an integer result.

```
{ Standard File Package ------------------------------------------}

 Procedure SFPutFile
   (
    where : Point;
    prompt : Str255;
    origName : Str255;
    dlgHook : ProcPtr;
    var reply : SFReply
   );

    Inline
   $3F3C, $0001,       {move.w #1, -(sp)}
   $A9EA;       {_Pack3}

 Procedure SFPPutFile
   (
    where : Point;
    prompt : Str255;
    origName : Str255;
    dlgHook : ProcPtr;
    var reply : SFReply;
    dlgID : Integer;
    filterProc : ProcPtr
   );

    Inline
   $3F3C, $0003,       {move.w #3, -(sp)}
   $A9EA;       {_Pack3}

 Procedure SFGetFile
   (
    where : Point;
    prompt : Str255;
    fileFilter: ProcPtr;
    numTypes : Integer;
    typeList : Ptr;
    dlgHook : ProcPtr;
    var reply : SFReply
   );

    Inline
   $3F3C, $0002,       {move.w #2, -(sp)}
   $A9EA;       {_Pack3}

 Procedure SFPGetFile
   (
    where : Point;
    prompt : Str255;
    fileFilter : ProcPtr;
    numTypes : Integer;
    typeList : Ptr;
```

```
    dlgHook : ProcPtr;
    var reply : SFReply;
    dlgID : Integer;
    filterProc : ProcPtr
   );

    Inline
  $3F3C, $0004,       {move.w #4, -(sp)}
  $A9EA;       {_Pack3}

{ International Utilities Package ----------------------------------}

 Function IUGetIntl
   (
    theID : Integer
   ) : Handle;

    Inline
  $3F3C, $0006,       {move.w #6, -(sp)}
  $A9ED;       {_Pack6}

 Procedure IUSetIntl
   (
    refNum : Integer;
    theID : Integer;
    intlParam : Handle
   );

    Inline
  $3F3C, $0008,       {move.w #8, -(sp)}
  $A9ED;       {_Pack6}

 Procedure IUDateString
   (
    dateTime : LongInt;
    longFlag : DateForm;
    var result : Str255
   );

    Inline
  $4267,       {clr.w -(sp)}
  $A9ED;       {_Pack6}

 Procedure IUDatePString
   (
    dateTime : LongInt;
    longFlag : DateForm;
    var result : Str255;
    intlParam : Handle
   );

    Inline
  $3F3C, $000E,       {move.w #14, -(sp)}
  $A9ED;       {_Pack6}

 Procedure IUTimeString
   (
    dateTime : LongInt;
    wantSeconds : Boolean;
    var result : Str255
   );

    Inline
  $3F3C, $0002,       {move.w #2, -(sp)}
  $A9ED;       {_Pack6}
```

```
  Procedure IUTimePString
    (
     dateTime : LongInt;
     wantSeconds : Boolean;
     var result : Str255;
     intlParam : Handle
    );

     Inline
    $3F3C, $0010,       {move.w #16, -(sp)}
    $A9ED;       {_Pack6}

  Function IUMetric : Boolean;

     Inline
    $3F3C, $0004,       {move.w #4, -(sp)}
    $A9ED;       {_Pack6}

  Function IUMagString
    (
     aPtr, bPtr : Ptr;
     aLen, bLen : Integer
    ) : Integer;

     Inline
    $3F3C, $000A,       {move.w #10, -(sp)}
    $A9ED;       {_Pack6}

  Function IUMagIDString
    (
     aPtr, bPtr : Ptr;
     aLen, bLen : Integer
    ) : Integer;

     Inline
    $3F3C, $000C,       {move.w #12, -(sp)}
    $A9ED;       {_Pack6}

{ Binary-Decimal Conversion Package ---------------------------------}

  Function StringToNum
    (
     theString : Str255
    ) : LongInt;

     Inline
    $205F,       {move.l (sp)+, a0}
    $3F3C, $0001,       {move.w #1, -(sp)}
    $A9EE,       {_Pack7}
    $2E80;       {move.l d0, (sp)}

  Procedure NumToString
    (
     theNum : LongInt;
     var theString : Str255
    );

     Inline
    $205F,       {move.l (sp)+, a0}
    $201F,       {move.l (sp)+, d0}
    $4267,       {clr.w -(sp)}
    $A9EE;       {_Pack7}
```

Lawrence D'Oliveiro

•••

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: Bug in THINK Pascal? My bug?

czychi@bernina.ethz.ch.UUCP (Gary Czychi) writes:
> Hi there,

Hi. I tried to send mail, but my mailer choked.

```
>  .
>  .
>  Writeln('Pick a point');
>  repeat
>   GetMouse(Horiz, Vert);
>  until Button;
>
>  MoveTo(firstHoriz, firstVert);
>  .
>  .
>
```

> But whenever I use this loop, it seems as if every single statement until
> the end of the program or until another reapeat..until loop is passed
> without execution!

By this do you mean that the text "Pick a point" never appears in the text window? If so, beat; it sounds like something is seriously scrambled on your disk (i. e. your project file, your THINK Pascal application, or your system folder---it's not a bug in LSP, because the exact same fragment compiles and executes fine on my system, which is identical to yours).

However, if the text *does* appear but the program doesn't pause to get the mouse, beware that if your button is still held down from some previous action (like a previous repeat/until button), this repeat loop will terminate immediately. When using constructs like this, it's always wise to preface them with

```
repeat
until (not button)
```

which will simply pause the program until it's certain that the user is no longer holding down the button. Then the next repeat/until button is guaranteed to cycle until the user clicks the button *for that loop* (rather than for some previous one).

I hope that's the problem. If not, try removing the objects from your project and rebuilding it, reinstalling THINK, and finally, reinstalling your system.

> The only possibility for me to achieve a correct behavior is, when I insert
> a 'stop' in the line directly after the loop and use 'go' until the end of
> the program. Then everything works as expected.
>
> I'm using THINK Pascal 3.0 on a SE/30 with 6.0.5. Maybe I've found a bug??
>
> Thanks a lot for any help.
>
> Gary T. Czychi University of St.Gallen, Switzerland

Nicholas Jackiw

•••

From: minow@mountn.dec.com (Martin Minow)
Subject: Re: A possible bug in THINK C 4.0.1

In article <425@spot.wbst128.xerox.com>, rainero.wbst@xerox.com (EmilRainero) asks about a possible odd-address error in Think C in coderoughly organized as:

```
char   something;
Str255 string;
```

One thing you might check is whether you're passing the address of "something" to a Toolbox variable that expects the address of a Byte or Boolean. Both of these are, suprisingly, 16-bit quantities. As a result, if "something" is at an odd-address, your program will crash in that toolbox routine.

Guess how I discovered this?

Martin Minow

•••

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: >32k arrays (in Think Pascal)

spellbinder@uwav1.u.washington.edu writes:

> Here is a compiler question for those programmers out in net-land.
>
> type ArrayRecord = record
> arraySize : longint;
> signal : array [0..1] of real;
> end;
> ArrayRecordPtr = ^ArrayRecord;
> ArrayRecordHdl = ^ArrayRecordPtr;
>
> The procedure knows how much data there is by getting the arraySize variable.
>
> Now I'm writing this code in THINK Pascal version 2.03. It works fine when the
> amount of memory dedicated to the signal array is less than 32K, but when it
> gets bigger than this, then things start to mess up. So basically, for array
> sizes less than 8K data points, everything's fine. My idea is that THINK Pascal
> has a problem getting a value at an offset greater than 32K from the start
> of the variable. I have verified my idea by changing signal to become
> an array of double and an array of extended and each time the procedure
> craps out when the data exceeds 32K. My feeling is that the compiler is not
> generating the correct code to access values beyond 32K.
>
> My questions are this:
>
> 1) Is there another explanation for the problem?
>
> 2) Are there any work arounds in THINK Pascal version 2.03?
>
> 3) Will THINK Pascal 3.01 avoid the problem?
>
> Thanks in advance.
>
> Blair Zajac Zajac@phast.phys.washington.edu

Your diagnosis is correct: Pascal uses the 68K's "Address Register Indirect With Displacement" addressing mode to access array data, and this mode's displacement (the offset) is limited to a sign- extended 16-bit displacement integer. Thus, you're limited to 32K arrays.

[A side note: in that it's sign-extended, it'd be possible to access 64K worth of data, if your base address was the arrayStart+32K, I think. Not that any Pascal compiler I've seen does this, of course...]

The work around is the same for THINK 2 and 3--you'll have to calculate your own offset. The best way to do this is with a bit of inline assembler:

```
type RealPtr=^real;

function MyData(Where:ArrayRecordHdl; Index:integer):realPtr;

inline $3017, $c0fc, $0004, $206f, $0002, $2050, $d1c0, $2f48, $0006, $5c4f;
```

To access your data, where formerly you used

```
ArrayRecordHdl^^[50]
```

now use

```
MyData(ArrayRecordHdl,50)^
```

Note that the third word of the in-line is $0004: this is the hardwired size of a single element of data in byte (1 REAL=4 bytes). The code compiles as:

```
MOVE.W   (A7),D0      ; Get the index off the stack
MULU.W   #4,D0        ; Multiply by the element size (32bit result)
MOVEA.L  2(A7),A0     ; Get a handle to the array
MOVEA.L  (A0),A0      ; Dereference it
ADDA.L   D0,A0        ; Add it to the calculated offset
MOVE.L   A0,6(A7)     ; Put resulting ptr back on the stack
ADDQ.W   #6,A7        ; and throw away arguments
```

Be sure to change $0004 to SizeOf(elem) if your elements ever change from being real. This code generates is a bit less efficient than Pascal's native array indexing, but it gets you past the 32K problem. Note the complete lack of range checking...hope it's okay.

---
jackiw@cs.swarthmore.edu

●●●

From: chewy@apple.com (Paul Snively)
Subject: Re: Multiple Inheritance for HandleObjects in C++

In article <15132@reed.UUCP> bowman@reed.UUCP (Eric Bowman) writes:

> I've just discovered, to my horror, that you can't create HandleObjects
> with multiple base classes in MPW C++. Are there any work arounds for this?
> Anyone know why this is the case?

Larry Rosenstein and Amanda Walker have already done an excellent job of answering the question Eric poses, but I'd like to point out something else:

Since Eric is using HandleObjects, he apparently isn't concerned about Object Pascal compatibility. This means that he's probably concerned about fragmenting the heap, which standard C++ objects, being based on malloc and free, tend to do.

Andy Shebanow, a former engineer here in MacDTS, wrote a very good article in "D e v e l o p" describing a class that he created called "PtrObject." The idea was to provide all of the functionality of standard C++ but to use NewPtr and DisposPtr to manage memory instead of malloc and free. Perhaps Eric would benefit from using Andy's PrtObject class.

Just a thought...

Paul Snively

●●●

Chapter 5 Dialogs & the Dialog Manager

Enter/Return in Modeless dialog 82
Informational dialog help (drawdialog stuff with code) 82
Password a'la AppleShare (Dialog Filter with code) 83
How can I draw contents of modeless dialog? 84
how can I draw contents of modeless dialog? 85
Floating Point #'s as Strings in Dialogs using Think C 3.0 85
Floating Point #'s as Strings in Dialogs using Think C 3.0 86
Query- Double lines on default button...How? 87
Query- Double lines on default button...How? 87
Password Dialogs--what is the best way to do them? 88

From:Ben Cranston <zben@Trantor.UMD.EDU
Subject: Re: Enter/Return in Modeless dialog

> ...IM-I p. 417 has some pretty good detail on how to interpret events
> in modeless dialogs. What [one] usually [does] is to call IsDialogEvent
> each time through the event loop. If it returns TRUE, go ahead and do any
> special processing of keys...

Some time ago I reported a gotcha about having to trap off activate events before IsDialogEvent in order to properly hilight the scroll bar of a List Manager list embedded in a modeless dialog. I got bit by another gotcha last week. If a modeless dialog event is frontmost, Multifinder events (like suspend and resume) are also considered "Dialog Events".
Thus, you have to check for THEM before calling IsDialogEvent...

The documentation on I-416/7 is still exactly correct:

'If theEvent is an activate or update event for a dialog window, a mouse-down event in the content region of an active dialog window, OR ANY OTHER TYPE OF EVENT WHEN A DIALOG WINDOW IS ACTIVE, IsDialogEvent returns TRUE...'

Emphasis added. ANY type of event. Caveat Codor.
--
Ben Cranston <zben@Trantor.UMD.EDU>

•••

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: Re: Informational dialog help (drawdialog stuff with code)

In article <1850@neoucom.UUCP> sam@neoucom.UUCP (Scott A. Mason) writes:

>I would like to display an informational dialog box that contains
>predefined static text items only. While this box is on the screen, I want
>to call a few procedures in my C program. When the procedures finish, I
>want the box to go away. Sounds pretty simple.

```
Handle GetDIHandle(i)Integer i;{
    short theType;
    Handle theHandle;
    Rect theRect;

    GetDItem(thePort, i, &theType, &theHandle, &theRect);
    return theHandle;
}

DialogPtr gMesgDia;

BeginMessage(){
    GrafPtr savePort;

    GetPort(&savePort);
    if(NIL == (gMesgDia = GetNewDialog(resID, NIL, (WindowPtr) -1L))){
        return;
    }
    SetPort(gMesgDia);
    SetIText(GetDIHandle(textITEM), (StringPtr) "\pYour message here.");
    ShowWindow(thePort);
    DrawDialog(thePort);
    SetPort(savePort);
}

EndMessage(){
    if(NIL != gMesgDia){
        DisposDialog(gMesgDia);
```

```
        gMesgDia = NIL;
    }
}
```

Now, just say:

```
BeginMessage();
... do your work ...
EndMessage();
```

You might also look at your event loop, and see if you are callingmisDialogEvent() and DialogSelect() there. You should be. See Inside Mac for more info on those two calls.

--- David Phillip Oster -- No, I come from Boston. I just work

●●●

From: blob@apple.com (Brian Bechtel)
Subject: Re: Password a'la AppleShare (Dialog Filter with code)

> I kept this code from the last time that this question was asked.
> >From: matthews@eleazar.dartmouth.edu (Jim Matthews)
> >Subject: Re: suppress display of password entry
>
> In article <536@sdacs.ucsd.EDU> wade@sdacs.ucsd.EDU (Wade Blomgren) writes:
> >
> >What is the best (easiest) way to allow entry of a password on the screen
> >while suppressing the display of the actual characters in the
> >edittext item?

It actually isn't very difficult to intercept the key events and keep a hidden copy of the password string. It isn't necessary to remember any context since you have access to the TextEdit record, and it tells you what the current selection is. The following routine is a filter for a name/password dialog box. It displays bullets (ala AppleShare) and stores the real password in a global string, pwStr. It also handles hitting return or enter.

```
{ signonFilter -- dialog filter for doSignon, hides password }

FUNCTION signonFilter (dp : DialogPtr;
            VAR theEvent : EventRecord;
            VAR itemHit : integer) : boolean;
    CONST
        nameItem = 3;
        passwordItem = 4;
        bs = $08;
        tab = $09;
        cr = $0D;
        enter = $03;
        larrow = $1C;
        rarrow = $1D;
        uparrow = $1E;
        downarrow = $1F;
    VAR
        dpeek : DialogPeek;
        theChar : char;
        theStr : Str255;
        selStart, selEnd : integer;
        h : Handle;
        itemType : integer;
        box : Rect;
BEGIN
    signonFilter := false;
    dpeek := DialogPeek(dp);
    IF ((theEvent.what = keydown) OR (theEvent.what = autoKey)) THEN
        IF (dpeek^.editField = passwordItem - 1) THEN
        BEGIN
```

```
            theChar := char(BitAnd(theEvent.message, charCodeMask));
            selStart := dpeek^.textH^^.selStart;
            selEnd := dpeek^.textH^^.selEnd;
            CASE ord(theChar) OF
                bs :                { Backspace }
                    BEGIN
                        IF selEnd = selStart THEN  { back over a character }
                        BEGIN
                            IF selStart > 0 THEN
                             pwStr := concat(copy(pwStr,1, selStart - 1),
                                             copy(pwStr, selStart + 1,
                                              length(pwStr) - selStart));
                        END
                        ELSE            { delete the selection }
                            pwStr := concat(copy(pwStr, 1, selStart),
                             copy(pwStr, selEnd + 1,
                                   length(pwStr) - selEnd));
                    END;
                cr, enter :     { Return or Enter -- treat as "OK }
                    BEGIN
                        itemHit := ok;
                        signonFilter := true;
                    END; { cr, enter }
                tab, uparrow, downarrow, rarrow, larrow :
                    ;     { just pass on tabs & arrows }
                OTHERWISE   { "normal" character }
                    BEGIN        { remember character, insert a bullet }
                        pwStr := concat(copy(pwStr, 1, selStart),
                         theChar,
                         copy(pwStr, selEnd + 1, length(pwStr) - selEnd));
                        theEvent.message :=
                         BitAnd(theEvent.message, $FFFFFF00) + ord('%');
                    END; { normal character }
            END; { case ord(theChar) of }
        END { in password field }
        ELSE     { not in password field -- still check for cr, enter }
            CASE BitAnd(theEvent.message, charCodeMask) OF
                cr, enter :
                    BEGIN
                        itemHit := ok;
                        signonFilter := true;
                    END; { cr, enter }
                OTHERWISE
                    ;
            END; { case BitAnd }
END; { signonFilter }
```

--Brian Bechtel blob@apple.com "My opinion, not Apple's"

•••

From: kk@mcnc.org (Krzysztof Kozminski)
Subject: Re: How can I draw contents of modeless dialog?

In article <24766@unix.cis.pitt.edu> er225711@unix.cis.pittsburgh.edu (M Kikuchi) writes:

> >I'm trying to implement a modeless dialog using complete DLOG and
> >DITL resources, but the contents of the item list are not drawn

You need to use IsDialogEvent/DialogSelect calls. Your event loop should look something like:

```
   if (GetNextEvent(everyEvent,& CurrEvent)) {
      if (IsDialogEvent(&CurrEvent)) {
    /* You may want to intercept some events here */
```

```
if (DialogSelect(&CurrEvent, &theDialog, &itemHit)) {
  /* Process the event/item that got hit */
}
  } else {
/* Handle non-dialog events */
  }
```

DialogSelect will process the update/activate events for you.

For more details, see IM vol.1, pages somewhere around 400,if I remember correctly.

KK
--
Kris Kozminski kk@mcnc.org

●●●

From: kazim@Apple.COM (Alex Kazim)
Subject: Re: how can I draw contents of modeless dialog?
Summary: Tucked away in Inside Mac
Keywords: Dialog

In article <24766@unix.cis.pitt.edu> er225711@unix.cis.pittsburgh.edu (M Kikuchi) writes:

> >I'm trying to implement a modeless dialog using complete DLOG and
> >DITL resources, but the contents of the item list are not drawn

Actually, both _ModalDialog and _DialogSelect make calls to, yes,_DrawDialog (see p. I-418). I remember having trouble about four years ago, and spending all afternoon trying to find this call (it's nicely tucked away.)

Alex Kazim, Apple Computer

●●●

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: Floating Point #'s as Strings in Dialogs using Think C 3.0

isler@grad1.cis.upenn.edu () writes:

> > I am writing a program in Think C 3.0 in which I would like to convert
> > floating point numbers to strings and place them in EditText items in
> > a dialog box.
> > sprintf(str, "%3.1f", value);
> >
> > Any suggestions?
> > S-K ISLER

As mentioned previously, you'll need a Pascal string instead of a c-string. You may be better served by using the Toolbox, instead of C, to do your conversions for you. SANE implements the calls:

```
procedure Str2Dec(s:DecStr;var Index:integer; var d:Decimal;
                  var validPrefix:boolean);

procedure Dec2Str(f:Decform; d:Decimal; var s:DecStr);

procedure CStr2Dec(s:CStrPtr;var index:integer; var d:Decimal;
                  var validPrefix:boolean);
```

which will do all the necessary conversions for you. In addition to providing the appropriate types to your task, SANE is script-manager compatible, which may or may not be true of THINK C's sprintf. This means that if your application is for public use, it will format (and accept formatted numbers) appropriately to the country in which it's in use.

See your SANE interface files for declarations of the individual types.

-----Nicholas Jackiw [jackiw@cs.swarthmore.eduljackiw@swarthmr.bitnet]-----

•••

From: stoms@castor.ncgia.ucsb.edu (David Stoms)
Subject: Re: Floating Point #'s as Strings in Dialogs using Think C 3.0

In article <25837@netnews.upenn.edu> isler@grad1.cis.upenn.edu () writes:

> >I am writing a program in Think C 3.0 in which I would like to convert
> >floating point numbers to strings and place them in EditText items in
> >a dialog box.
>
> >The result is that garbage is printed in the dialog instead of the
> >converted string. The above conversion method works fine when I am
> >displaying floats as strings in windows, but it does not work here.

```
typedef char[32] FStr;

Double2String(double num, int digits, FStr string)
    {
    char   bcdNum[12];
    int        i, j;
    char   c;

    if (!string) Debugger();

/* I believe SANE does this too */
    asm {
        fmove.x    num,fp0
        fmove.p    fp0,bcdNum{#17}
        }

    j = 1;
    c = bcdNum[3];
    if (bcdNum[0] & 0x8000) string[j++] = '-';
    string[j++] = (c & 0xf) + '0';
    string[j++] = '.';

    digits = 16 - digits;
    if (nbtwn(digits, 0, 16)) Debugger();

    for (i=4; i <= 11 - (digits>>1); i++) {
        c = bcdNum[i];
        string[j++] = (c>>4 & 0xf) + '0';
        string[j++] = (c & 0xf) + '0';
        }
    j -= digits % 2;
    /* while (string[j-1] == '0') j--; */
    if (string[j-1] == '.') j--;

    string[j++] = 'e';
    string[j++] = (bcdNum[0] & 0x4000) ? '-' : '+';

    i = false;
    if ( (bcdNum[0] & 0xf) != 0 ) {
        string[j++] = (bcdNum[0] & 0xf) + '0';
        i = true;      /* significant digit */
        }
    if ( i || (bcdNum[1]>>4 & 0xf) != 0) {
        string[j++] = (bcdNum[1]>>4 & 0xf) + '0';
        i = true;
        }
    if ( !i && bcdNum[1] & 0xf == 0)
```

```
        j -= 4;
    else
        string[j++] = (bcdNum[1] & 0xf) + '0';

    string[0] = j-1;
    }
```

Josh.

•••

From: lippin@brahms.berkeley.edu (The Apathist)
Subject: Re: Query- Double lines on default button...How?

I find that an inactive PICT item does a wonderful job of highlighting the default button. It has two advantages over drawing the ring "by hand" -- it uses no code, and it updates whenever necessary.

I even use this method in alerts -- although the dialog manager draws the outline for you, it doesn't update it if your window gets covered (by MultiFinder or a screen saver). A similar problem applies to the icons for NoteAlert, CautionAlert, and StopAlert, and in this case can be solved with an appropriate ICON item.

--Tom Lippincott

•••

From: keith@Apple.COM (Keith Rollin)
Subject: Re: Query- Double lines on default button...How?

In article <3486@adobe.UUCP> hawley@adobe.UUCP (Steve Hawley) writes:

> >In article <9883@odin.corp.sgi.com> myoung@joker.sgi.com (Mark Young) writes:
> >>that describes how to make a button with the double lines that indicates
> >>the default selection. I found a comment indicating that item #1 was the
> >>default selection, but my "ok" button, which is item #1 doesn't appear with
> >>the telltale double lines?
> >>
> >>any clues?
> >
> >Bad news: The bold outline does NOT get drawn for you. You must do that
> >yourself.
> >
> >Inside Macintosh Volume I has some code to do that. It boils down to getting
> >the bounding rect of the item, InsetRect(&boundingRect, -4, -4), PenSize(2, 2),
> >and then calling FrameRoundRect(...) to draw it.

Following is the routine that DTS will be using in future versions of our sample code. Note that it DOES NOT use the (16, 16) method recommended by Inside Mac, which is why I'm posting it here.

```
PROCEDURE OutlineButton(button: UNIV ControlHandle);

{Given any control handle, this will draw an outline around it.  This is used for the
default button of a window.  The extra nice feature here is that I'll erase the outline for
buttons that are inactive.  Seems like there should be a Toolbox call for getting a
control's hilite state. Since there isn't, I have to look into the control record myself.
This should be called for update and activate events.

 The method for determining the oval diameters for the roundrect is a  little different than
that recommended by Inside Mac. IM I-407 suggests that you use a hardcoded (16,16) for the
diameters.  However, this only looks good for small roundrects. For larger ones, the outline
doesn't  follow the inner roundrect because the CDEF for simply buttons doesn't use (16,16).
Instead, it uses half the height of the button as the  diameter. By using this formula, too,
our outlines look better.

 WARNING: This will set the current port to the control's window.}
```

```
CONST

kButtonFrameSize= 3;      { button frameUs pen size }
kButtonFrameInset= - 4;{ inset rectangle adjustment around button }

VAR
    theRect:          Rect;
    curPen:           PenState;
    buttonOval:       integer;

BEGIN
    IF button <> NIL THEN BEGIN
        SetPort(button^^.contrlOwner);
        GetPenState(curPen);
        PenNormal;
        theRect := button^^.contrlRect;
        InsetRect(theRect, kButtonFrameInset, kButtonFrameInset);
        buttonOval := (theRect.bottom - theRect.top) DIV 2;
        IF (button^^.contrlHilite = kCntlActivate) THEN
            PenPat(black)
        ELSE
            PenPat(gray);
        PenSize(kButtonFrameSize, kButtonFrameSize);
        FrameRoundRect(theRect, buttonOval, buttonOval);
        SetPenState(curPen);
    END;
END;
```

--
Keith Rollin --- Apple Computer, Inc. --- Developer Technical Support

●●●

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: Password Dialogs--what is the best way to do them?

tj@cs.ucla.edu (Tom Johnson) writes:

> My big question is: How do I know when the user is entering text into
> the Password text edit item (as opposed to the UserName item)? The event
> record gives me no indication of which item is selected (or does it?),
> the itemHit variable doesn't tell me, and I don't where to find out.

There's a field in the DialogRecord called editField, which according to I-408 "is one less than the item number of the current editText item, or -1 if there's no editText item in the dialog." So use this and bob's your uncle.

Do beware of tabs, though. ModalDialog interprets these separately, as keys to switch from the current editText item to the next. This switching probably happens after your filterProc and before your itemHit. Make sure you ignore tabs, or handle them appropriately.

> Tom Johnson UCLA Computer Science Department

-----Nicholas Jackiw [jackiw@cs.swarthmore.eduljackiw@swarthmr.bitnet]-----

●●●

Chapter 6 File Manager

Finding amount of free disk space? 90
How to get pathname (with Code) 90
User items in SFdialog 91
SFGetFolder (with Code) 91
Repeat SetVol query 95
SFGetFolder 95
INIT getting it's file name... (with Code) 96
INIT getting it's file name...(with Code) 97
opendir() and readdir()(enumerate dir code in pascal) 97
Please help w/ PBSetCatInfo 99
FSRead hangs on the serial driver 99
FSRead hangs on the serial driver 100
The Prefered Way to Refer to Files... 100
SFReply.vRefNum -> PathName
(pathname from vRefnum and Back in C) 101
Create a folder code (in Pascal) 103
How do I get a pathname from a working directory? (full Code) 105
How to get full pathname? 107
"SetVol-ing" to a folder in the System Folder... 107
Data Fork Use 108
Data Fork Of Currently Running APPL 108
Data Fork Use 108

From: blob@apple.com (Brian Bechtel)
Subject: Re: Finding amount of free disk space?

In article <4995@uhccux.uhcc.hawaii.edu> mikem@uhccux.uhcc.hawaii.edu (Mike Morton) writes:

> I've got a document open which a user specified with SFGetFile, so all I
> have to identify it is a vRefNum and a name. I'd like to find out how
> much free space there is on the drive on which the document is stored
> (i.e., how much more can I expand the document?).
>
> It looks like I want to use GetVInfo, which takes a drvNum as a
> parameter. If so, how do I map a vRefNum to the drvNum containing it?
> If not, what's the right way?

Call PBHGetVInfo, passing an ioCompletion of NIL, ioNamePtr of NIL, ioVRefNum of the vRefNum gotten from SFGetFile, and ioVolIndex of 0. You'll get back ioVFrBlk and ioVAlBlkSiz. Multiply those together to get the free bytes on the disk. Or, if you're lazy, pass ioVDrvInfo to your high level call of GetVInfo, and you'll get the multiplication done for you.

--Brian Bechtel blob@apple.com

●●●

From:Al Evans-
Subject: How to get pathname (with Code)

In article <850zebolskyd@yvax.byu.edu> zebolskyd@yvax.byu.edu writes:

>If you want the _user_ to be able to see the pathname (and find their way to
>the file) you can use PBGetWDInfo to get the dirID from your working
>directory, then PBGetCatInfo to get that directory's parent's dirID, and
>so on until to get to dirID=2, which will be the root. You get the names
^^^^^^^

Although this seems to be true, it doesn't seem to be documented :-(

>along the way from ioNamePtr. That's also how I get the volume name for
>the volumename/dirID/filename resource I use to save a file's location
>for future _machine_ use. I can send you Modula-2 source code if you want,
>but you would probably be better off making your _own_ mistakes instead
>of trying to find mine.
>
>--Lyle D. Gunderson zebolskyd@yvax.byu.edu CIS: 73760,2354

There are MANY good reasons for never using full pathnames in a Mac application. But I, too, have stumbled across situations where it was absolutely necessary. I found that the technique recommended above works, but was unable to find any guarantee that the root directory would ALWAYS have dirID=2. As far as I can tell, the following routine does not rely upon anything undocumented:

-----cut about here--------

```
{
Returns full pathname to folder specified by startID in thePath, where
startID is the vRefNum/wdRefNum returned by SFGetFile or SFPutFile
}

    PROCEDURE GetCurrentPath (startID : INTEGER; VAR thePath : Str255);
        VAR
            tempName : Str255;
            vParams : CInfoPBRec;
            theError : OSErr;
    BEGIN
        thePath := '';
        tempName := '';
        WITH vParams DO
            BEGIN
                ioCompletion := NIL;
                ioNamePtr := @tempName;
```

```
                ioVRefNum := startID;
                ioFDirIndex := -1;
                ioDrDirID := 0;
                REPEAT
                    theError := PBGetCatInfo(@vParams, FALSE);
                    IF (theError = noErr) THEN
                        BEGIN
                            ioDRDirID := ioDRParID;
                            thePath := concat(tempName, ':', thePath);
                            tempName := '';
                        END;
                UNTIL (theError <> noErr);
            END;
    END;
```

--Al Evans--

•••

From: Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com
Subject:User items in SFdialog

In article <2426@tekno.chalmers.se> d83_sven_a@tekno.chalmers.se (SVEN AXELSSON) writes:
>Hi.

Hello.

> >I want to put a user-item into my custom SFGetFile dialog. How do I get the
> >user-item procedure connected? I can't do the usual GetItem - SetItem stuff
> >since SFPGetFile will read the dialog template for me. Is there a way of
> >handling this, short of writing my own filterProc - something I do NOT
> >want to do.

No. But I don't see what's wrong with writing your own filter proc. Just type in the function definition from Inside Mac and pass the name of the function to SFPGetFile. It's certainly no harder than writing a user item drawing procedure.

Your item filter procedure will get passed a -1 item for initialization. This is the time to do the GetDItem-SetDItem soft-shoe to bind the user item drawing procedure. The entire function shouldn't take more than ten lines if this is all you're doing.
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

•••

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: **SFGetFolder (with Code)**

In article <709@maytag.waterloo.edu> jb@aries5.UUCP () writes:
> >Has anybody written a SFGetFolder/SFPutFolder routine -- What I want to be
> > able to do is select or create a folder in a nice manner.

This is the second most common question here, after "How do I get a full file name?" Apple, are you listening? How about a couple of new traps in System 7.0?

Anyway, the answer is yes. LSC source code at end of message.

> >The problem that I have with the implementations that I have seen of
> >selecting a folder i.e. 'Set Transfer Directory...' in Telnet is that it
> >is not intuitive whether I am selecting the directory of the folder
> >that is hilighted or the directory that I am currently in.

That's very perceptive. The solution is not to allow the user to click on the "Use Folder" (or whatever) button when a folder is selected, and to put a sentence in the dialog explaining which folder is being dealt with just to make sure there's no confusion. Hardly anyone does this in practice, but everyone should.

Here goes with the code.

```
static Boolean good, noSys, needWrite, allowFloppy, allowDesktop;
static SFReply reply;

pascal Boolean
FolderFilter(pb)
FileParam *pb;
{
	return true;
}

pascal short
FolderItems(item, dlog)
short item;
DialogPtr dlog;
{
	if (item == 2) {
		good = true;
		item = 3;
	}
	return item;
}

pascal void
FolderEvents(dialog, event, item)
DialogPtr dialog;
EventRecord *event;
short *item;
{
	ControlHandle ch;
	short type;
	Rect r;
	HVolumeParam vp;

	/* disable if a directory is selected in the list */
	GetDItem(dialog, 2, &type, &ch, &r);
	if (reply.fType) {
		HiliteControl(ch, 255);
		return;
	}

	/* get information on the volume */
	vp.ioNamePtr = (StringPtr)0;
	vp.ioVRefNum = -SFSaveDisk;
	vp.ioVolIndex = 0;
	if (PBHGetVInfo(&vp, false))
		HiliteControl(ch, 255);
	else if (vp.ioVSigWord != 0x4244)		/* HFS? */
		HiliteControl(ch, 255);
	else if (vp.ioVDRefNum >= 0 || vp.ioVDrvInfo == 0)  /* ejected? */
		HiliteControl(ch, 255);
	else if (needWrite && (vp.ioVAtrb & 0x8080))		/* locked? */
		HiliteControl(ch, 255);
	else if (!allowFloppy && vp.ioVDRefNum == -5)		/* floppy? */
		HiliteControl(ch, 255);
	else if (!allowDesktop && CurDirStore == 2)		/* desktop? */
		HiliteControl(ch, 255);
	else if (noSys && CurDirStore == vp.ioVFndrInfo[0]) /* blessed? */
		HiliteControl(ch, 255);
	else	HiliteControl(ch, 0);
}
```

```
GetFolder(name, volume, folder, writeable, system, floppy, desktop)
char *name;
short *volume;
long *folder;
Boolean writeable, system, floppy, desktop;
{
        short oldvol = -SFSaveDisk;
        long oldfolder = CurDirStore;
        Point where;
        SetPt(&where, 55, 55);
        good = false;
        if (*volume && *folder) {
                SFSaveDisk = -*volume;
                CurDirStore = *folder;
        }
        needWrite = writeable, noSys = !system, allowFloppy = floppy;
        allowDesktop = desktop;
        SFPGetFile(where, (char *)0, FolderFilter, -1, 0, FolderItems,
                  &reply, 14, FolderEvents);
        if (!good) return false;
        *volume = -SFSaveDisk;
        *folder = CurDirStore;
        FullFileName(name, "", -SFSaveDisk, CurDirStore);
        SFSaveDisk = -oldvol;
        CurDirStore = oldfolder;
        return true;
}
```

Note -- you need dialog 14 for this to work. This is a slightly hacked version of Standard File. Here's a Rez listing made by DeRez:

```
resource 'DLOG' (14, "Standard File for a Folder") {
    {55, 78, 312, 439},
    dBoxProc,
    invisible,
    noGoAway,
    0x0,
    14,
    ""
};

resource 'DITL' (14, "Standard File for a Folder") {
    {   /* array DITLarray: 11 elements */
        /* [1] */
        {33, 507, 51, 587},
        Button {
            enabled,
            "Open"
        },
        /* [2] */
        {135, 256, 153, 336},
        Button {
            enabled,
            "Use Folder"
        },
        /* [3] */
        {161, 256, 179, 336},
        Button {
            enabled,
            "Cancel"
        },
        /* [4] */
        {40, 247, 62, 348},
        UserItem {
            disabled
```

```
        },
        /* [5] */
        {69, 256, 87, 336},
        Button {
            enabled,
            "Eject"
        },
        /* [6] */
        {95, 256, 113, 336},
        Button {
            enabled,
            "Drive"
        },
        /* [7] */
        {40, 15, 185, 246},
        UserItem {
            enabled
        },
        /* [8] */
        {40, 229, 185, 246},
        UserItem {
            enabled
        },
        /* [9] */
        {124, 251, 125, 339},
        UserItem {
            disabled
        },
        /* [10] */
        {97, 606, 198, 702},
        StaticText {
            disabled,
            ""
        },
        /* [11] */
        {196, 15, 246, 345},
        StaticText {
            disabled,
            "Move until ^0 is shown in the small rect"
            "angle at the top, above the list of file"
            "s and folders.  Then click \"Use Folder\"."
        }
    }
};
```

The "^0" is filled in by the caller of GetFolder, using ParamText. Granted, that's not the cleanest solution, but GetFolder already has too many parameters. If you are only using this for one thing in your software, then you can always just fill it in in the dialog item in the resource file instead.

No flames from symbolic-constant freaks. It's just as easy to change a number in a routine as it is to change it in #defines, if it only appears once. And the dialog items are not going to change; their order is mandated by Standard File.

I do apologize for the use of globals, but as all Standard File hackers know, the system doesn't give you the ability to do without them easily. If your filter procs were passed a pointer to the reply record, you could embed it in a structure and use its pointer as a structure pointer, but it doesn't get passed. And touching the dialog refCon causes an explosion. If you really can't cope with globals, you're stuck with using a button refCon or something, and that's a bit of a mess. (Also, I don't think I'd figured out that kind of devious solution back when I wrote this.)

The one great omission in this code is a "New" button. One day I'll get around to adding it.
--

•••

From: tim@hoptoad.uucp (Tim Maroney)

Subject: Re: Repeat SetVol query

From article <3260@carthage.cs.swarthmore.edu>, by jackiw@cs.swarthmore.edu (Nick Jackiw):
> You can't SetVol(...) to the working directory returned by SFGetFile
> under HFS.

In article <2802@hub.UUCP> 6600pete@hub.UUCP writes:
> Sure you can. I've done it lots. What error code are you getting back?

In article <3273@carthage.cs.swarthmore.edu> jackiw@carthage (Nick Jackiw) writes:
>No error. Just no results. Seems to me the following program, which
>calls SFGetFile thrice, should on the last instance bring up the GetFile
>dialog in the same directory/volume as a file was chosen in on the first
>instance. Got that?

Ah-hah! This once again goes to show how important it is to list your symptoms in detail. If you'd said this, you would have been besieged by messages pointing out that Standard File and the current HFS directory have almost nothing to do with each other.

If you have to set the directory for Standard File (and this *is* sometimes a valid operation, though indiscriminate use can confuse the user -- remember that power users in particularly frequently "blind type" to Standard File and expect it to work predictably) you do it
with a pair of low-memory globals called SFSaveDisk (negative of volume reference number) and CurDirStore (directory id). I demonstrated this technique in the code I just posted on hacking Standard File to select folders. It's more or less documented in Inside Mac IV, chapter 15. Set these globals to the necessary values before your call to Standard File.

To do this with a working directory reference number, like the one you get from Standard File, you have to convert the working directory into a volume/folder pair. You do this like so:

```
SFVolDir(wd, volume, folder)
short wd, *volume;
long *folder;
{
        WDPBRec pb;
        char name[72];
        pb.ioNamePtr = (StringPtr)name;
        pb.ioVRefNum = wd;
        pb.ioWDIndex = 0;
        pb.ioWDProcID = 0;
        pb.ioWDVRefNum = 0;
        PBGetWDInfo(&pb, false);
        if (pb.ioResult) return pb.ioResult;
        *volume = pb.ioWDVRefNum;
        *folder = pb.ioWDDirID;
        return noErr;
}
```

--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: keith@Apple.COM (Keith Rollin)
SubjectRe: SFGetFolder

In article <9741@zodiac.ADS.COM> jtn@ads.com (John Nelson) writes:
>In article <709@maytag.waterloo.edu> jb@aries5.UUCP () writes:
>>Has anybody written a SFGetFolder/SFPutFolder routine -- What I want to be
>> able to do is select or create a folder in a nice manner.
>>
>>The problem that I have with the implementations that I have seen of
>>selecting a folder i.e. 'Set Transfer Directory...'

MacDTS has released a sample program that shows you how to do this, among other tricks with StdFile. You can get this from APDA, BBS's and finer FTP sites everywhere, including ours. Basically, it involves writing a file filter procedure that always returns TRUE. Here is the relevant code:

```
FUNCTION FoldersOnly(p:ParmBlkPtr):BOOLEAN;

{ Normally, folders are ALWAYS shown, and aren't even passed to        }
{ this file filter for judgement. Under such circumstances, it is      }
{ only necessary to blindly return TRUE (allow no files whatsoever).   }
{ However, Standard File is not documented in such a manner, and       }
{ this feature may not be TRUE in the future. Therefore, we DO check   }
{ to see if the entry passed to us describes a file or a directory.    }

    BEGIN
        FoldersOnly := TRUE;
        IF BTst(p^.ioFlAttrib,4) THEN FoldersOnly := FALSE;
    END;
```

Hope this helps,

-
Keith Rollin --- Apple Computer, Inc. --- Developer Technical Support

•••

From: unknown
Subject: Re: HFS Help

This point is clearly covered in Inside Mac v. 4. the "vRef" of a folder is really a "wRef" a working directory reference. It only lives for a short time. You cannot depend on it from session to session. There is another number, a 4-byte dirId that is the permanent identifier of the folder, even if the folder is renamed. There is a system call that takes a wRef and a real volume reference and returns the dirId. PBHOpen() takes a volume Id, a dirId, and a file name and opens the file. To store a file descriptor from run to run, you need a triple: the volume name, the dirId, and the file name. To be really safe, you need the full path name too, so you can use it on a remote file system that doesn't really support dirIds. You need the volume name as a string since it may get a different number depending on the order of the mounting of the disks at boot time.

•••

From: mm5l+@andrew.cmu.edu (Matthew Mashyna)
Subject: Re: INIT getting it's file name... (with Code)

Ando Sonenblick: sonenbli@oxy.edu writes:
> >...is there any way
> >for an INIT to get the name of the file it is in? I need to later open
> >the resource fork of the file and I want the user to be able to change the
> >name of the file; hence, my INIT needs to be able to read in the file name.

Check the latest issue of MacTutor: Writing INITs in Think C, by J.Peter Hoddie.

```
findMyName(name)
Str255 name;
{
        FCBPRec p;

        p.ioCompletion = 0;
        p.ioRefNum = CurResFile();
        p.ioVRefNum = 0;
        p.ioNamePtr = (StringPtr) name;

        PBGetFCBInfo(&p,false);

        BlockMove( p.ioNamePtr,&name, 1 + *(char *)(p.ioNamePtr) );
}
```

Matt Mashyna

•••

From: svc@well.UUCP (Leonard Rosenthol)
Subject**Re: INIT getting it's file name...(with Code)**

In article <4020@helios.ee.lbl.gov> beard@ux1.lbl.gov (Patrick C Beard) writes:
>In article <56505@tiger.oxy.edu> sonenbli@oxy.edu (Andrew D. Sonenblick) writes:
>>
>> Ok, this is a quick one but it is very important: is there any way
>>for an INIT to get the name of the file it is in? I need to later open
>>the resource fork of the file and I want the user to be able to change the
>>name of the file; hence, my INIT needs to be able to read in the file name.
>>Please inform me or the whole net of any solution. Thanks.

I don't know what all this mess is about searching the System Folder, etc. as that is a REAL PAIN, and it also doesn't provide help for when you use some INIT Manager that supports subFolders...So here is the piece of code that I use for getting not only my INIT's name, but also the vRefNum (OK, WDRefNum) where I am.

```
void GetInitInfo(myName, myVRefNum)
Str255  *myName;
int     *myVRefNum;
{
        FCBPBRec        p;
        int                     dummy;

        p.ioCompletion = 0;
        p.ioRefNum = CurResFile();
        p.ioFCBIndx = 0;
        p.ioVRefNum = 0;
        p.ioNamePtr = (StringPtr)myName;
        dummy = PBGetFCBInfo(&p, FALSE);

        BlockMove(p.ioNamePtr, &myName, (long)p.ioNamePtr[0]);
        dummy = GetVol(NIL, myVRefNum);
}
```

[Pascal conversion is left as an exercise for the reader if required!]

--
Leonard Rosenthol | GEnie : MACgician

•••

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: **opendir() and readdir()(enumerate dir code in pascal)**

h+@nada.kth.se (Jon W{tte) writes:
> I simply want to get the names of the files in adirectory.
> No, I am no novice to mac programming, but I've let libraries
> handle file I/O for me until now. I looked through IM IV, but
> the HFS chapter is less than crystal clear on this point...
>
> Now, how do I search {a | the current} directory for file
> names (and maybe types...) ? Any code pieces or hints are
> welcome !

Try this.

```
program enumerateDir;

var
```

```
    ourParam: ParmBlkPtr;    {For finding ChainMail file in System Folder}
    ourWDParam: WDPBPtr;  {For opening a working directory}
    mailBoxAddr, mailFileName: str255;
    fileNum: integer; {For indexing all SysFolder files}
    theErr: OSErr;        {Misc. OS Call error code}

begin
    MailBoxAddr:='Saturn:System Folder:'; {**whatever**}
    with ourWDParam^ do
        begin
            ioCompletion := nil;
            ioNamePtr := @MailBoxAddr;
            ioVRefNum := 0;
            ioWDProcID := 0;
            ioWDDirId := 0;
        end;

    if PBOpenWD(ourWDParam, false) <> noErr then
        begin
            writeln('Too many working directories open!');
            readln
        end;

    writeln('Here goes!');

    fileNum := 1;            {Begin examining all files}

    with ourParam^ do
        begin
            ioCompletion := nil;
            ioNamePtr := @mailFileName;
            ioVRefNum := ourWDParam^.ioVRefnum;
            mailFileName := '';
        end;

    repeat

        ourParam^.ioFDirIndex := fileNum;
        theErr := PBGetFInfo(ourParam, false);

        if theErr = noErr then
    begin
writeln(mailFileName, ' ', ourParam^.ioFlFndrInfo.fdType, ' ',
ourParam^.ioFlFndrInfo.fdCreator);
    end;

        fileNum := succ(fileNum);
    until theErr <> noErr;

end.
```

Sorry about all the tabs...vi seems to have a wider setting than LSP.

This is a fragment cut from working code. Reading it once, nothing looks spurious, but there may be references to the particular system in which it was formerly embedded. The last writln in the program should list the filename, file type, and creator field, as per your request.

Nicholas Jackiw [jackiw@cs.swarthmore.eduljackiw@swarthmr.bitnet]

•••

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: **Please help w/ PBSetCatInfo**

>In article <30340@shemp.CS.UCLA.EDU> tj@oahu.cs.ucla.edu (Tom Johnson) writes:
>> error=PBGetCatInfo(&myInfoRec,false);
>> if (error != noErr) SysBeep(10);
>> [flags set, etc.]
>> error=PBSetCatInfo(&myInfoRec,false);
>> if (error != noErr) Debugger();
>>
>>When I run this code, no error is detected on the PBSetCatInfo call and
>>the CInfoPBRec appears to be correctly filled in, but PBSetCatInfo
>>returns error=FFD5 a fnfError (file not found). Does anybody have
>>any idea why? This has been driving me crazy for hours!!

In article <37560@apple.Apple.COM> keith@Apple.COM (Keith Rollin) writes:
>I think that what is happening is that your ioDirID field is being changed from
>MyCurID on the PBGetCatInfo call. Then, when you try the PBSetCatInfo call,
>you are using a DirID different from the original one.

Yes. PBGetCatInfo changes the dirID it returns, apparently to the file ID. (Yet another place where this pointless feature is a pain in the ass.) It is necessary to save the dirID before calling PBGetCatInfo, then to reset the ioDirID field to this saved value before calling PBSetCatInfo. And of course, this is completely undocumented, unless you count the mysterious two-headed arrow before ioDirID in the description of the PBGetCatInfo routine.
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: cc100aa@prism.gatech.EDU (Ray Spalding)
Subject: Re: **FSRead hangs on the serial driver**

In article <7770@unix.SRI.COM> gil@ginger.sri.com (Gil Porat) writes:
[regarding FSRead with the serial driver]
>1) Why does the second FSRead hang?

Because it will not return until it gets the exact number of characters you asked it to read.

>2) Doesn't FSRead return as much as it can?

No, see above.

>3) Is there a way to get the serial driver and/or FSRead to time out?

The way to accomplish this is to use asynchronous I/O (in the sense of IM vol IV, not in the sense of async data communications). You start an asynchronous read request, then go about your business processing events and so on. Periodically, you check the request block for completion: 0 == normal completion; 1 == still in progress; anything else == error. If you want to time it out, you have to watch the time yourself (using TickCount).

Caveat: Be sure to call SystemTask (or WaitNextEvent) periodically so that the serial driver will get a slice of CPU.

Some fragments of the code I use (MPW C):

```
ParamBlockRec commInPB;
static unsigned char commInChar;
static short commInPort = -6;

CommStartIn() /* Queue serial port input request and return */
{
    commInPB.ioParam.ioCompletion = NULL;
    commInPB.ioParam.ioVRefNum = 0;
    commInPB.ioParam.ioRefNum = commInPort;
    commInPB.ioParam.ioBuffer = &commInChar;
    commInPB.ioParam.ioReqCount = 1; /* ask for one character only */
    commInPB.ioParam.ioPosMode = 0;
    PBRead(&commInPB,true);
}
```

```
CommDisp() /* Check for serial port input completion */
{
    int rc;

    SystemTask();
    rc = commInPB.ioParam.ioResult;
    if (rc == 1) return; /* or check for timeout here */
    if (rc != noErr) { /* process errors */ return;}
    /* ... process one input character ... */
    CommStartIn(); /* queue request for next character */
    return;
}
```

> >4) Does FSRead need an EOT to return?

No, EOT is passed through to you like any other character.

--
Ray Spalding

●●●

From: chesley@goofy.apple.com (Harry Chesley)
Subject: Re: FSRead hangs on the serial driver

In article <4713@hydra.gatech.EDU> cc100aa@prism.gatech.EDU (Ray Spalding) writes:

> >1) Why does the second FSRead hang?
> >
> Because it will not return until it gets the exact number of characters
> you asked it to read.

More exactly: it will read until it gets at least as many characters as you ask for (more is OK).

It sounds like you're experiencing either buffer overflow or baud rate mismatch. The default input buffer for the serial port is only 64 bytes. So 80 characters can easily overflow the buffer if you don't read it out fast. You can increase the buffer size by calling SerSetBuf (but be sure and unset the buffer by calling SerSetBuf with a length of zero before exiting the application or you'll get horrible crashes).

Alternatively, the baud rates may not be set right. Therefore, you send enough characters at one speed, but they come through as fewer gibberish characters at the other speed.

In article <4713@hydra.gatech.EDU> cc100aa@prism.gatech.EDU (Ray Spalding) writes:

> >3) Is there a way to get the serial driver and/or FSRead to time out?
> >
> The way to accomplish this is to use asynchronous I/O (in the
> sense of IM vol IV, not in the sense of async data communications).
> You start an asynchronous read request, then go about your business processing
> events and so on. Periodically, you check the request block for completion:
> 0 == normal completion; 1 == still in progress; anything else == error.
> If you want to time it out, you have to watch the time yourself (using
> TickCount).

Even simpler, use SerGetBuf to find out how many input characters are available, and do a synchronous call but only when there are enough characters available.

●●●

From: brecher@well.sf.ca.us (Steve Brecher)
Subject: Re: The Prefered Way to Refer to Files...
Keywords: File Manager

In article <38534@cornell.UUCP>, wayner@svax.cs.cornell.edu (Peter Wayner) writes:

> I've been coding up an application which needs to keep a list of files

> and their locations on the disk. This list needs to be stable from boot
> to boot and even between restores from backup. There are several ways
> this can be done:
>
> 1) Full Path name
> 2) file name and DirId

2) should be volume name, file name, and DirID.

> My questions are;
>
> Are there any other problems with pathnames?

The other problem with pathnames is that they may exceed 255 bytes in length, which may make them difficult to use.

The best approach would be to store both (1) and (2), with code to handle the 256+ length problem mentioned above. When accessing the file, use (2); if that works, update (1). If (2) doesn't work, use (1) and update (2). However, few if any products actually do this; my own current products use (2) only. One of them (PowerStation) has a command to search all online disks for "lost" files, as typically occurs after an initialize/restore.

> How can I convert a DirId into a vRef which I can then use with [FSOpen]?
> Or more generally, how can I open a file with a name and a DirId?

As noted above, you will nead a volume specifier in addition to file name and DirID. Instead of using FSOpen, you can use PBHOpen. But to answer the question: you would create a WD with OpenWD, and then pass a WDRefNum instead of a VRefNum to FSOpen. Since WDs occupy a system-wide table of limited size, you would clean up afterwards with CloseWD.

> What about System 7.0? The tech note mentions that file id numbers will
> be the best when System 7.0 comes around because they will be stable
> even after renaming.

Right, but it ain't here yet; also, you will probably want your application to run under earlier systems. In a couple of years or so it will be feasible to require System 7 or later, just as circa 1988 it became feasible to require the Mac Plus feature set (introduced in early 1986).
--
brecher@well.sf.ca.us (Steve Brecher)

•••

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: SFReply.vRefNum -> PathName (pathname from vRefnum and Back in C)

The Pascal source posted ignores errors and string lengths, and so it can easily crash your machine if the path name runs to more than 255 characters or if an error leaves directoryName with a large length byte. Here's some C code to do the same thing (MPW C 3.0, but since it was originally written in LSC 3.0 it shouldn't be hard to convert back). FullFileName should be passed an empty string in tail if you just want the name of a folder.

```
SFVolDir(short wd, short *volume, long *folder)
{   WDPBRec pb; unsigned char name[72];
    pb.ioNamePtr = (StringPtr)name;
    pb.ioVRefNum = wd;
    pb.ioWDIndex = 0;
    pb.ioWDProcID = 0;
    pb.ioWDVRefNum = 0;
    PBGetWDInfo(&pb, false);
    if (pb.ioResult) return pb.ioResult;
    *volume = pb.ioWDVRefNum;
    *folder = pb.ioWDDirID;
    return noErr;
}
```

```
FullFileName(StringPtr outname, StringPtr tail, short volume, long dirID)
{   StringHandle name; Str255 text, volname; HVolumeParam hvp;
    CInfoPBRec di; long size; short err;

    /* extract the volume name */
    hvp.ioNamePtr = volname;
    hvp.ioVRefNum = volume;
    hvp.ioVolIndex = 0;
    PBHGetVInfo((HParmBlkPtr)&hvp, false);
    if (hvp.ioVSigWord == 0xd2d7) { outname[0] = 0; return noHFSerr; }

    /* create and initialize the name handle */
    if (tail) {
        if (err = PtrToHand(tail + 1, &name, tail[0])) return err;
        size = tail[0];
    }
    else { if ((name = NewHandle(0)) == 0) return MemError();
        size = 0;
    }

    /* now start extracting the dirs and prepending them to
     * the handle
     */
    for ( ; dirID != 2; dirID = di.dirInfo.ioDrParID) {
        text[0] = 0;
        di.dirInfo.ioNamePtr = text;
        di.dirInfo.ioVRefNum = volume;
        di.dirInfo.ioFDirIndex = -1;
        di.dirInfo.ioDrDirID = dirID;
        PBGetCatInfo(&di, false);
        text[++text[0]] = ':';
        SetHandleSize(name, size += text[0]);
        BlockMove((Ptr)*name, (Ptr)(*name) + text[0], size - text[0]);
        BlockMove((Ptr)text + 1, (Ptr)*name, text[0]);
    }

    /* prepend the volume name onto the handle */
    volname[++volname[0]] = ':';
    SetHandleSize(name, size += volname[0]);
    BlockMove((Ptr)*name, (Ptr)(*name) + volname[0], size - volname[0]);
    BlockMove((Ptr)volname + 1, (Ptr)*name, volname[0]);

    /* copy and delete the handle */
    if (size > 255) {
        DisposHandle(name);
        SysBeep(12);
        outname[0] = 0;
        return tooBigErr;
    }
    outname[0] = size;
    BlockMove((Ptr)*name, (Ptr)outname + 1, size);
    DisposHandle(name);
    return noErr;
}
```

And on the theory that you might want to convert the name back some day, here's a routine I use for that. Notice that it will actually create the directory if it doesn't exist now. If this isn't what you want, throw away the error recovery code.

```
FullToFolder(StringPtr name, short *volume, long *folder)
{   CInfoPBRec dirInfo; HVolumeParam hvp; short i;
    dirInfo.dirInfo.ioNamePtr = name;
    dirInfo.dirInfo.ioDrDirID = 0;
    dirInfo.dirInfo.ioVRefNum = 0;
```

```
    dirInfo.dirInfo.ioFDirIndex = 0;
    if (PBGetCatInfo (&dirInfo, 0) == noErr) {
        if (dirInfo.dirInfo.ioFlAttrib & 0x10) {
            *folder = dirInfo.dirInfo.ioDrDirID;
            hvp.ioNamePtr = name;
            for (i = 0; i < name[0]; i++)
                if (name[i+1] == ':')
                    { name[0] = i + 1; break; }
            hvp.ioVolIndex = -1;
            hvp.ioVRefNum = 0;
            PBHGetVInfo ((HParmBlkPtr)&hvp, 0);
            *volume = hvp.ioVRefNum;
        }
        else {  *volume = 0, *folder = 0; return -1; }
    }
    else if (dirInfo.dirInfo.ioResult == fnfErr
         || dirInfo.dirInfo.ioResult == dirNFErr)
    {   dirInfo.dirInfo.ioNamePtr = (StringPtr)name;
        dirInfo.dirInfo.ioDrDirID = 0;
        dirInfo.dirInfo.ioVRefNum = 0;
        if (PBDirCreate((HParmBlkPtr)&dirInfo, false) == noErr) {
            *volume = dirInfo.dirInfo.ioVRefNum;
            *folder = dirInfo.dirInfo.ioDrDirID;
        }
        else {  *volume = 0, *folder = 0;
            return dirInfo.dirInfo.ioResult;
        }
    }
    else    { *volume = 0, *folder = 0; return dirInfo.dirInfo.ioResult; }
    return noErr;
}
```

--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From:Bill Stackhouse
Subject:Create a folder code (in Pascal)

{The other day, someone posted a source for creating folders along a pathname.

I have updated that concept to include:

- specify a full path name
- use the volume name specifed in the path name
- use the file name spedifed in the path name
- if one or more directories in path already exist, use them
- if a specified directory has the same name as an existing file, stop
- allow specification of a volume name as a volref num
- allow specification of a starting directory as a dirID
- return the volref num and the dirID of the final directory created

Bill Stackhouse
bills@XAIT.Xerox.COM}

```
unit UCreateFolder;

interface

function CreateFolder (pName: string;
      var pVRefNum: Integer;
      var pDirID: Integer): OSErr;

{CreateFolder - create all folders and files in a pathname. }
```

```
{ pName - vol:dir:dir: ... :file }
{         if vol ommited, will use pVRefNum }
{         if file ommited, only the directories will be created. }
{pVRefNum - input: if pName does not specify vol, the volRefNum. }
{         output: the ioVRefNum used to create the directories and file. }
{pDirID - input: if not 0, the dirID where to start creating directories. }
{         output: the ioDirID of the last directory in the list. }
{}
{ Result codes }
{   aabbb where aa = the vol/dir/file which caused the }
{       error and bbb = the error. }
{   example:  }
{       input a:b:c  or  :b:c   with result = -2048 is 'c' is dup file name. }
{       input a:b   with result = -0035 is 'a' not mounted. }
{ noErr   - ok }
{ nsvErr  - vol name not mounted }
{ dupFNErr  - either file exists in directory or needed to }
{         create folder with same name as existing file. }
{         Existing folders with same name are not errors. }

implementation

  function CreateFolder (pName: string;
       var pVRefNum: Integer;
       var pDirID: Integer): OSErr;

    var
      i: Integer;
      count: Integer;
      err: OSErr;
      theParms: HParamBlockRec;
      theCat: CInfoPBRec;
      theName: string[31];
  begin
    theParms.ioCompletion := nil;
    theParms.ioResult := 0;
    theParms.ioVRefNum := pVRefNum;
    theParms.ioDirID := pDirID;
    theCat.ioCompletion := nil;
    theCat.ioResult := 0;
    theCat.ioVRefNum := theParms.ioVRefNum;
    theCat.ioFDirIndex := 0;
    theCat.ioDrDirID := theParms.ioDirID;
    i := Pos(':', pName);
    theName := Copy(pName, 1, i - 1);
    Delete(pName, 1, i);
    if Length(theName) > 0 then {set ioVRefNum to correct vol}
      begin
        theName := Concat(theName, ':');
        theParms.ioNamePtr := @theName;
        theParms.ioVolIndex := -1;
        err := PBHGetVInfo(@theParms, FALSE);
        if err = nsvErr then
          begin
            CreateFolder := err;
            Exit(CreateFolder);
          end;
        theParms.ioDirID := pDirID;
      end;
    count := 0;
    repeat
      count := count + 1;
      i := Pos(':', pName);
      if i = 0 then
        begin
          theName := Copy(pName, 1, Length(pName));
```

```
          Delete(pName, 1, Length(pName));
          theParms.ioNamePtr := @theName;
          err := PBHCreate(@theParms, FALSE);
        end
      else
        begin
          theName := Copy(pName, 1, i - 1);
          Delete(pName, 1, i);
          theParms.ioNamePtr := @theName;
          err := PBDirCreate(@theParms, FALSE);
          if err = DupFNErr then
            begin
              theCat.ioVRefNum := theParms.ioVRefNum;
              theCat.ioNamePtr := @theName;
              theCat.ioDrDirID := 0;
              err := PBGetCatInfo(@theCat, FALSE);
              if not BitTst(@theCat.ioFlAttrib, 3) then {found file with same n}
                err := DupFNErr;
              theParms.ioDirID := theCat.ioDrDirID;
            end;
        end;
    until (Length(pName) = 0) or (err <> noErr);
    pVRefNum := theParms.ioVRefNum;
    pDirID := theParms.ioDirID;
    if err <> noErr then
      err := err - (count * 1000);
    CreateFolder := err;
  end;    {CreateFolder}

end.
```

•••

From: Apple DTS
Subject: How do I get a pathname from a working directory? (full Code)

```
(** PathNameFromDirID ************************************************)
(*
(*  Given a DirID and real vRefnum, this routine will create and return the
(*  full pathname that corresponds to it. It does this by calling PBGetCatInfo
(*  for the given directory, and finding out its name and the DirID of its
(*  parent. It the performs the same operation on the parent, sticking ITS
(*  name onto the beginning of the first directory. This whole process is
(*  carried out until we have processed the root directory (identified with
(*  a DirID of 2.
(*
(*  NOTE: This routine is now A/UX friendly. A/UX likes sub-directories
(*        separated by slashes in a pathname. This routine automatically
(*        uses colons or slashes as separators based on the value of the
(*        global gHasAUX.  This global must be initialized correctly for
(*        this routine to do its thing. However, because of this dependancy
(*        on the idiosyncracies of file systems, generating full pathnames
(*        for other than display purposes is discouraged; it's changed in
(*        the past when A/UX was implemented, and it may change again in
(*        the future it support for other file systems such as ProDOS,
(*        MS-DOS, or OS/2 are added.
(*
(********************************************************************)

  FUNCTION PathNameFromDirID (DirID:longint; vRefnum:integer):str255;

    VAR
      Block : CInfoPBRec;
      directoryName, FullPathName : str255;
```

```
    BEGIN
      FullPathName := '';
      WITH block DO BEGIN
        ioNamePtr := @directoryName;
        ioDrParID := DirId;
      END;

      REPEAT
        WITH block DO BEGIN
          ioVRefNum := vRefNum;
          ioFDirIndex := -1;
          ioDrDirID := block.ioDrParID;
        END;
        err := PBGetCatInfo(@Block,FALSE);

        IF haveAUX THEN BEGIN
          IF directoryName[1] <> '/' THEN BEGIN
            { If this isn't root (i.e. "/"), append a slash ('/') }
            directoryName := concat(directoryName, '/');
          END;
        END ELSE BEGIN
          directoryName := concat(directoryName,':');
        END;
        fullPathName := concat(directoryName,fullPathName);

      UNTIL (block.ioDrDirID = 2);

      PathNameFromDirID := fullPathName;
    END;


(** PathNameFromWD ******************************************)
(*
(*  Given an HFS working directory, this routine returns the full pathname
(*  that corresponds to it. It does this by calling PBGetWDInfo to get the
(*  VRefNum and DirID of the real directory. It then calls PathNameFromDirID,
(*  and returns its result.
(*
(************************************************************************)

  FUNCTION PathNameFromWD(vRefNum:longint):str255;

    VAR
      myBlock : WDPBRec;

    BEGIN

      {
      { PBGetWDInfo has a bug under A/UX 1.1.  If vRefNum is a real vRefNum
      { and not a wdRefNum, then it returns garbage.  Since A/UX has only 1
      { volume (in the Macintosh sense) and only 1 root directory, this can
      { occur only when a file has been selected in the root directory (/).
      { So we look for this and hardcode the DirID and vRefNum. }

      IF (haveAUX) AND (vRefNum = -1) THEN BEGIN

        PathNameFromWD := PathNameFromDirID(2,-1);

      END ELSE BEGIN

        WITH myBlock DO BEGIN
          ioNamePtr := NIL;
          ioVRefNum := vRefNum;
          ioWDIndex := 0;
          ioWDProcID := 0;
        END;
```

```
    { Change the Working Directory number in vRefnum into a real vRefnum }
    { and DirID. The real vRefnum is returned in ioVRefnum, and the real }
    { DirID is returned in ioWDDirID. }

      err := PBGetWDInfo(@myBlock,FALSE);

      WITH myBlock DO
        PathNameFromWD := PathNameFromDirID(ioWDDirID,ioWDVRefnum)
    END;
  END;
```

•••

From: lefty@twg.com (David N. Schlesinger)
Subject: Re: How to get full pathname?

In article <PETE.90May12102735@tone.rice.edu> pete@tone.rice.edu (Pete Keleher) writes:

> For an application that I have been working on, I'd like to be able display
> either just the filename, or the complete path name.

Assuming that you are starting out with a filename and a directory ID, the way to proceed is as follows:

1] Call GetCatInfo with ioFDirIndex = -1, and with your base directory ID in ioDrDirID. When the call returns, you'll have the parent directory ID in ioDrParID and the name of the initial directory in the buffer pointed to by ioNamePtr. Prepend the name to your file name with a colon in between.

2] Call GetCatInfo again, the same way, but put the parent directory's ID into ioDrDirID. Again, prepend the name returned to the string you're building, with a colon between the new piece and what you've already built. Repeat this step until you get back an ioDrParID of 2; this indicates the root. Do it one last time to get the name of the root directory. Prepend this (again followed by a colon) to the string you've built so far.

Voila! A full pathname! Be careful, though. Full pathname can easily exceed 255 characters. This can cause major problems if you're using Str255s instead of c strings...

Hope this helps...

David N. Schlesinger

•••

From: tim@efi.com (Tim Maroney)
Subject: Re: "SetVol-ing" to a folder in the System Folder...

In article <101186@tiger.oxy.edu> sonenbli@oxy.edu (Andrew D. Sonenblick) writes:

> Anyway, to the business at hand: I have tried many combinations of
>code, many variations of paramBlocks, etc. and yet I still cannot set the
>volume to a specific folder (I am not using SFGetFile()).
> What I do want to do is an effective "SetVol" to a folder (the name of
>which I know) which is in the System Folder (the vRefNum of which I know) so
>that if I call "OpenResFile" for example the Mac would attempt to open a file
>which is inside the folder in question.

What you should do is take the SysEnvirons working directory reference number for the system folder, and pass it to PBGetCatInfo in the ioVRefNum field, with ioFDirIndex set to zero and ioNamePtr set to the folder name. The directory id of the folder you want will be returned in ioDrDirID. Now, armed with a working directory reference number and a directory id, you have to make a new working directory for the folder. First call PBGetWDInfo on the system folder WD reference number, to get the volume reference number (ioWDVRefNum). Then pass this volume reference number and the directory id from PBGetCatInfo to PBOpenWD. Finally you have a working directory reference number for the folder. You can pass this to SetVol or PBSetVol. Don't use PBHSetVol, even though it lets you leave out a step or two -- see the relevant tech note on why this call is usually dangerous.

See? It's a snap! Why, any technical summa cum laude from a good ivy league school could do it in no more than a few months! The beauty and elegance of the Macintosh file system have never been surpassed.

●●●

From: austing@Apple.COM (Glenn L. Austin)
Subject: Re: Data Fork Use

wolf@mel.cipl.uiowa writes:

> >I tried once before to get information on using the data fork for applications.
> >Has anyone had experience with this? Can someone suggest a good source for
> >information?

You can simply open the data fork. No big deal, but you DO have to get the name and location of the Application (PBGetFCBInfo, IM IV, pg. 179). The refNum of the resource file is the reference number returned from OpenResFile or CurResFile.

--
Glenn L. Austin

●●●

From: austing@Apple.COM (Glenn L. Austin)
Subject: Re: Data Fork Of Currently Running APPL

cl29+@andrew.cmu.edu (Cameron Christopher Long) writes:

> >I am working on a program that stores data in its data fork. Currently,
> >I am opening the data fork using FOPEN since I know the name of the APPL.
> >However, if someone renames the program, how will I know what to open?

Use PBGetFCBInfo (IM4, pg. 179) with the refCon of the application's resource fork as the file refCon. This gives you the name of the application, whether it is locked, where it is on disk, etc.

> >All suggestions are most appreciated,
> >Cameron Altenhof-Long

You're welcome.

--
Glenn L. Austin

●●●

From: rmh@apple.com (Rick Holzgrafe)
Subject: Re: Data Fork Use

In article <42652@apple.Apple.COM> austing@Apple.COM (Glenn L. Austin) writes:

> > wolf@mel.cipl.uiowa writes:
> >
> > >I tried once before to get information on using the data fork for applications.
> > > >Has anyone had experience with this? Can someone suggest a good source for
> > > >information?
> >
> > You can simply open the data fork. No big deal, but you DO have to get the
> > name and location of the Application (PBGetFCBInfo, IM IV, pg. 179).
> > The refNum of the resource file is the reference number returned from
> > OpenResFile or CurResFile.

Perhaps easier is to get the app's name by calling GetAppParms (IM II, pg. 58) and using that and the default volume refnum (zero, or call GetVol at startup and save it) to do the open.

You can write in the data fork if you like, but then your app won't run right on locked or read-only volumes such as CD-ROMs and some file servers. Reading is no problem.

(Tried to mail this notion, but it bounced. Sorry to trouble the net.)

Rick Holzgrafe | {sun,voder,nsc,mtxinu,dual}!apple!rmh

●●●

Chapter 7 List Manager & LDEFS

List Manager advice 112
List Bummers Revisited (simple custom ldef with code) 112
List Bummers Revisited (and Debugging Help) 114
List definitions and fonts... 114
Cdevs and the List Manager 115

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: List Manager advice

rdd@walt.cc.utexas.edu (Robert Dorsett) writes:

> I'm attempting to use the List Manager, from within a dialog, to implement a
> one-column list of text cells. The length of the list can vary, with items
> My problem is that I'm unable to get the cell ID of the currently-selected
> item. I'm convinced I'm adding data to the list correctly list correctly
> (dataBounds for the list looks okay), but if I try LLast to get the point, it
> doesn't work. If, on the other hand, I use LGetSelect, the numbers are too
> high (but not by much, which raises the maddening question of whether I'm
> installing the data correctly).
>
> Robert Dorsett

I assume from your post that your list permits only one item to be selected at a time. Good.

```
begin {Modal List Dialog}
 theDLOG:=GetNewDialog(theDLOG_Id,nil,pointer(-1));
 InstallUserItem(theDLOG,List_ITEM,@DrawList);
 SetPort(theDLOG);

 {At this point, it's assumed the list already exists; you've done LNew}

 theList^^.SelFlags:=lOnlyOne;

 ShowWindow(theDLOG);
 LDoDraw(true,theList);

 repeat
  ModalDialog(@ModalProc,itemHit);
  if itemHit=List_Item then                {Translate double-clicks into Opens}
   if LClick(GlobalPt,0,theList) then itemHit:=Open_ITEM;
 until (itemHit=Open_ITEM) or (itemHit=Cancel_ITEM);

if itemHit=open_ITEM then
 begin {Figure out which item is selected; that's what we want to open}
  SetPt(aPt,0,0);
  junk:=LGetSelect(true,aPt,theList);
  aStr:='####################';
  oldLen:=length(aStr);
  LGetCell(ptr(longint(@aStr)+1),oldLen,aPt,theList);

{aStr now contains the text of your item to be opened; act appropriately}
 ...
 end;
end;
```

This scheme uses a global variable, globalPt, to record the coordinates of the last mousedown (that's the purpose of modalProc), so that our main loop, above, can send that point to the list manager for double-clicks. (Alternately, doubleClicks could be detected in the filterProc, which would obviate the need for a global.)

If this doesn't help, why don't you post your code?

--
Nick Jackiw

●●●

From: amanda@mermaid.intercon.com (Amanda Walker)
Subject: Re: List Bummers Revisited (simple custom ldef with code)

In article <cZVSJ9i00WB2Q63mcX@andrew.cmu.edu>, es2q+@andrew.cmu.edu (Erik Warren Selberg) writes:

> Is there any way to access
> variables in the main program from an LDEF, and if so, how??!

Well, they way I do it is to have a baby LDEF that, by default, just draws a string they way LDEF 0 does, but if the list's refCon is non-zero, treats it as a pointer to a function and calls it to do the drawing. All you have to do is make sure A5 is set correctly.

Here's the source for the LDEF in MPW C 3.0, taken in its entirety from known working code. It does contain "the cast from Hell," but hey :-).

```
----------------------
/*
    Simple Custom LDEF

    Amanda Walker, InterCon Corporation
    6 October 1988

    This is a very simple custom LDEF.  It acts just like LDEF 0 if the refCon
    field of the list is 0.  If it is non-zero, the LDEF treats it as the
    address of a routine to call to actually draw a list element.

*/

#include <Quickdraw.h>
#include <Lists.h>
#include <OSUtils.h>

#define nil ((void *) 0L)

pascal void STUBLDEF(lMessage, lSelect, lRect, lCell, lDataOffset, lDataLen,
            lHandle)
short lMessage;
Boolean lSelect;
Rect *lRect;
Cell *lCell;
short lDataOffset, lDataLen;
ListHandle lHandle;
{
    char *p;
    long savedA5;

    /* Make sure A5 is valid so the draw code can use its globals. */
    savedA5 = SetCurrentA5();

    p = (char *) 0x938;      /* HiliteMode for color QD */
    lMessage--;          /* no initialization */
    if (!lMessage--) {       /* draw the cell */
    if ((**lHandle).refCon)
        (*((pascal void (*)(Rect *, Cell *, short, short, ListHandle))
        (**lHandle).refCon))(lRect, lCell, lDataOffset, lDataLen,
                    lHandle);
    else {
        MoveTo(lRect->left + 5, lRect->top + 12);
        HLock((**lHandle).cells);
        DrawText(*((**lHandle).cells), lDataOffset, lDataLen);
        HUnlock((**lHandle).cells);
    }
    if (lSelect) {
        *p = *p & 0x7f;
        InvertRect(lRect);
    }
    } else if (!lMessage--) {       /* toggle highlighting */
    *p = *p & 0x7f;
```

```
    InvertRect(lRect);
    }
    SetA5(savedA5);      /* no closing code */
}
```

If you don't know how to build code resources with MPW C, take a look at the examples in the CExamples directory.

Hope this helps,

Amanda Walker InterCon Systems Corporation

●●●

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: List Bummers Revisited (and Debugging Help)

In article <cZVSJ9i00WB2Q63mcX@andrew.cmu.edu> es2q+@andrew.cmu.edu (Erik Warren Selberg) writes:
>ok... I finally fixed the clicking bug (by SetPorting every time I checked
>if the application was mine (& thus did something with it).

Great, send me a million dollars now.

>I'd like to put the data object I have (called TFile, an object with a bunch
>of neat stuff contained) into the list instead of using up 100 bytes a shot
>as I am now... I've tried putting it in using StripAddress, putting in the
>starting file (well, a pointer to the starting file) in the list's refCon and
>trying to access it from there, and nothing. Is there any way to access
>variables in the main program from an LDEF, and if so, how??! what am I doing
>wrong?

I don't know; once again, your report has been far too vague to allow any solid deductions or even founded speculations. "And nothing" is no more a report of a symptom that is "I don't feel so hot". You can store common data structures in the list's refCon or in the list's userHandle. These are then accessible by the LDEF through the list handle it is passed, and by the application through the list handle it is keeping track of. What's the problem? Does the refCon come out as zero or some other invalid value in the LDEF? Then you're not stashing the value correctly. The syntax would be "list^^.refCon = LONGINT(<whatever>);" if I remember my Pascal.

>also: is there any way to debug an LDEF (or any code resource) from LSP?
>this trial & error stuff is rather depressing!

Lots of ways. First, to debug on the Mac, you simply must learn a low level debugger, such as MacsBug or TMON. You can debug anything this way. David Oster just gave a description of the process. Briefly, place a Debugger instruction (0xa9ff) in your code at the point where you want to start watching the fun, and then step through. It helps to have a paper listing of your program so you can follow along better.

You can also track the general flow of execution in various other ways. If you're not sure a piece of code is getting executed, put a SysBeep in it. And so on.

If you simply can't live without a symbolic debugger, then you use a pass-through code resource. Your actual LDEF resource is six bytes, starting with the word for a JMP.L instruction with an absolute long operand (0x4ef9), and having the address of your definition procedure in the second and third words. You link the definition procedure (the LDEF code) with your application, and set up the six-byte LDEF resource with the address of the main LDEF routine before you create your list, making sure it's non-purgeable so it won't get refeshed from disk. Then you can put symbolic breakpoints in your LDEF to your heart's content. After it's debugged sufficiently, you can make it a real, separate LDEF again if you wish -- or you can leave it the way it is.
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: olson@bootsie.UUCP (Eric Olson)
Subject: Re: List definitions and fonts...

In article <4715@helios.ee.lbl.gov> beard@ux1.lbl.gov (Patrick C Beard) writes:

>In article <36618@cornell.UUCP> wayner@cs.cornell.edu (Peter Wayner) writes:
>#I just built a scrolling list of names in my window
>#and it works fine. I would like to change the font
>#of the list, though, to be something other than
>#the application font. I can't seem to find the
>#correct way to do this.
>#
>
>What I do is bracket all of my calls to the list manager that do drawing
>with calls to TextFont and TextSize with the appropriate font and size.
>Then, I restore the old font and size to whatever is needed. This
>seems to work.

If you change the font after the list has been drawn, you may wish to set the indent field of the List Record. The LDEF sets this during the init message, but then doesn't expect the font to change. Something like:

```
GetFontInfo(&fontinfo);
(*macList)->indent.h = 4;
(*macList)->indent.v = fontinfo.ascent;
```

These are the values that Apple's LDEF 0 chooses at list init time.

Note that you only need to change this if the font in the list changes dynamically, not if you just change it before calling LNew.

-Eric

●●●

From: ech@cbnewsk.att.com (ned.horvath)
Subject: Re: Cdevs and the List Manager

From article <13580@unix.SRI.COM>, by mxmora@unix.SRI.COM (Matt Mora):

> ...The whole idea of
> the update routine is to only draw what really needs to be updated. Not to
> invalidate the whole mess and redraw it.
>
> Does anybody know the real reason why the listmanager sometimes doesn't
> update correctly?

To add a bit more fuel to this thread: I've noticed that LUpdate does a ValidRect on the part of the window where the horizontal scrollbar is drawn -- even if the list has no horizontal scrollbar. I traced this one through the 6.0.5 PACK0 code: sure enough, it tests for the presence of a scrollbar, then (if there isn't one), branches around the draw routine to the ValidRect.

If LUpdate is only being called between BeginUpdate and EndUpdate, this doesn't matter. But if LUpdate is being called after resizing the list, this is annoying. The only workaround I've found is to explicitly call InvalRect after LUpdate. I have not checked to see if the same problem exists when there is no vertical scrollbar.

=Ned Horvath=

●●●

Chapter 8 Memory and the Memory Manager

ROM Unlocks handles
(hlock and movehi slows programs down w/code) 118
Completion routine questions and anwsers 118
Segmentation (tips on using unloadseg) 119
Tips on unloadseg (and a story) 120
Handles and Virtual Memory (rewriting the Mem Mgr) 121
Setting up multiple heaps 121
NewPtrclear,NewhandleClear 122
NewPtrclear,NewhandleClear 123
StripAddress and pointer arithmetic 124
StripAddress and pointer arithmetic 124
llci Trap dispatch Table 125
Identifying real handles (with Code) 125
Identifying real handles 126
Need large array on Think C. 126
Need large array on Think C. 127
Need large array on Think C. (explanation on how to do it) 127
Need large array on Think C. 128
Identifying real handles 129

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: Re: ROM Unlocks handles (hlock and movehi slows programs down w/code)

In article <3842@atr-la.atr.co.jp> alain@atr-la.atr.co.jp (Alain de Cheveigne) writes:
>But hassle it is. Is there a better way ?

Yes. What complexity, and slowness! (all those unnecessary MovHHi()s can really slow a program down.) I too use very similar data structrues: handles in the RefCons of my windows that themselves reference other handles.

If a field referenced by an O.S. call is small, copy it. If it is large, lock the owning handle during the call. Do not write your own routines to take pointers into handles, instead, if you must do such a thing, do it by passing a handle an an offset. (Often you can just pass the handle, since the routine knows its offset.)

Here is an example, from a program that uses a window handle which references two pixmaps. (I keep the current window handle in a global variable called "theDoc", by analogy with "thePort".) Notice, no need for any HLocks or MoveHHis.

```
/* DocDispose - discard our window
 */
DocDispose(){
    if(NIL != (**theDoc).newPix){
        if(NIL != (**(**theDoc).newPix).baseAddr){
            DisposPtr((**(**theDoc).newPix).baseAddr);
            (**(**theDoc).newPix).baseAddr = NIL;
        }
        DisposPixMap((**theDoc).newPix);
    }
    if(NIL != (**theDoc).oldPix){
        if(NIL != (**(**theDoc).oldPix).baseAddr){
            DisposPtr((**(**theDoc).oldPix).baseAddr);
            (**(**theDoc).oldPix).baseAddr = NIL;
        }
        DisposPixMap((**theDoc).oldPix);
    }
    DisposHandle((Handle) theDoc);
    DisposeWindow(thePort);
}
```

Actually, this does bring up a question. Inside Mac Vol 5 doesn't say precisely which fields are disposed when you call DisposPixMap().

--- David Phillip Oster

●●●

From:Rick Holzgrafe
Subject: **Completion routine questions and anwsers**

In article <6732@hubcap.clemson.edu> mikeoro@hubcap.clemson.edu (Michael K O'Rourke) writes:

> If i am in the background and enter the iocompletion routine, is it
> possible to get interrupted or swapped out halfway thru the completion
> routine? I am having some funky errors when running in the background
> under multifinder and can't seem to figure out what is going on.

Your completion routine may be called from a higher-than-normal interrupt level, but not at the highest possible level. A higher-level interrupt can, um, interrupt you. But this shouldn't bother you any more than *your* interrupt bothers whoever you interrupted. MultiFinder in particular won't swap processes without being deliberately called: don't make any "Event" calls (e.g. WaitNextEvent, GetNextEvent) and you won't be yielding control to MultiFinder.

I don't know what your specific problem is, but here's some general hints:

- Keep completion routines SHORT. Best is to quickly store results someplace where your main-line code can find them, and get out.

- Minimize your stack usage in completion routines. You don't know whose stack you're running in, or whose heap you'll trash if you overflow.
- You can't call the Memory Manager from a completion routine. This means you can't call ANY routine which may move or purge memory: see the appendix in Inside Mac for a list of these routines. (I bet you knew that one already. :-)
- You can make new asynchronous calls from a completion routine. (But beware of the pitfalls mentioned in these tips.) You cannot make synchronous calls from completion routines.
- Be aware that your completion routine can (in some cases) be called BEFORE your async PBControl call returns! Don't rely on having any time after the PBControl to get ready for the completion. And if you're making new async calls in your completion routines, beware of "recursion" and stack overflow.
- Remember that your completion routine can be interrupted by a higher-priority interrupt, and that it will itself be interruptingyour main-line code. You may need to protect critical code or data. If you do this (typically by temporarily forcing the interrupt level high), do it as rarely and as quickly as you can.
- Be sure your code segment containing the completion routine is loaded, locked, and unpurgeable. Don't unload it!
- Be careful about A5 (access to global variables, and calls to routines outside the segment). A5 is not guaranteed correct in a completion routine.
- Don't allocate your parameter blocks as local variables of routines which may return before the completion.

Hope this helps.

-
Rick Holzgrafe

●●●

From: lippin@spam.berkeley.edu (The Apathist)
Subject: Re: Segmentation (tips on using unloadseg)

Recently tim@hoptoad.UUCP (Tim Maroney) wrote:
>And sure, you can move things into smaller partitions this way, as long
>as you don't care about certain minor facts like keeping to the
>interface guidelines. User operations are supposed to begin taking
>effect instantaneously. If they require fetching in a large resource
>from the application file, then they are anything but instantaneous.

Remember, those segments aren't getting swapped out unless you needed the memory for something else. When your user is pushing the program to where this would matter, the difference between unloading and not unloading is the difference between running slowly and not running at all. I think it's clear which one the user wants.

> >You also ignored the function pointer issue. I received a letter from
> >a person at Apple that I respect and who is usually right, but not this
> >time. He says that you should put all function-pointer fetching
> >routines into the main code segment: that way, they will be jump table
> >pointers. This may work fine for code that uses one or two function
> >pointers and calls them all itself. However, just imagine passing a
> >function pointer into the jump table to the vertical retrace manager or
> >as a completion routine to the file manager. Can you say "intermittent
> >crash when A5 is not CurrentA5"? I knew you could.

Both of you got this one wrong. Any development system worth its salt, and certainly any I've used, will create function pointers as pointers into the jump table, precisely so that they're always valid. They're truly absolute pointers, so they don't care one whit about CurrentA5 (although your code might; that's the well-known gotcha of interrupt tasks, and has established solutions.)

[user-confusion caused by bugs that don't happen deleted]

>I prefer instant response and program reliability, as well as the
>ability to use function pointers freely in my code. There are really
>very few people who need to use more than two or three programs in
>conjunction with the Finder at once, and those people can get SIMMs
>pretty cheap these days. Real RAM hogs like MPW Shell and FullWrite
>may need to use it, but my programs rarely get over 250K.

Wow! You make it sound like having teeth pulled! It's not so difficult as all that.

Tom's three easy rules to a long and happy life using UnloadSeg:

1. Put the trap glue in your main segment.
2. Put your low memory emergency code in your main segment.
3. Put your main event loop in your main segment, and have it call UnloadSeg on all other segments, often.

Clever programmers can improve somewhat on this, depending on the structure of their particular programs. But this will work well, and your users would appreciate it, if somebody explained the difference to them.

After all, this is the computer for the rest of them.

--Tom Lippincott

•••

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: Tips on unloadseg (and a story)

I recently HAD to use UnloadSeg(). The marketing manager called me on one of the products I was writing and said, "Help, the program is now so big it will only allow 2400 records on a 1Meg MacPlus. The box says 2600. It will take six weeks to make new boxes and we need to ship NOW!"

So, I had to use unloadseg to get space when the user's files got large.

Here is the big problem: There MUST be space to load a required segment. If the o.s. can't find space, it will crash. You are making an innocent, arbitrary procedure call, and _kaboom_. Hard to debug and hard to fix.

Some work arounds:

1.) You can do a setjump in your main loop, and a longjump in your growzone routine. This should sort of work. Note though, your growzone function is called whenever anything needs memory including the o.s. doing a longjump from inside the o.s. is a good way to leave it in an inconstant state. For your own code, you can use Signal (see the tech notes,) and declare a cleanup routine that will get executed during the longjump, but it isn't easy to retrofit this kind of thing into an almost done 40,000 line program.

2.) Your development system will let you find out what segment (by number)each procedure is in. You can add a few:

```
if(NIL == GetResource('CODE', MAINDOCUPDATESEGNUM)){
    Error(IAMSORRYDAVE);
}
```

that is, the program makes a probe to see if it will be able to get the segment it needs to do a command. If it can't the user sees an error message. (This assumes the error message code is in the main segment.) This is technique I used. My programs are object oriented, so my message dispatcher can return the error code MESSAGEUNDELIVERABLENOTATHOME, if I send a message to an object whose code can not be read.

This has some problems: I have to manage my own symbols for my segments, which might get out of date as I rearrange my procedures. I wish there were a system call: CanLoadSeg(procedureName) which returns NIL or a handle to the segment. It always bothers me when I see a function with an obvious inverse that is not implemented. (Yes, I know I could write it, but would it work in System 7?)

My next program will use technique 1, unless you good folk can tell me a better way to do things.

I usually only unload segments in my main loop, but occasionally, when I need extra room (for example during a Save or a spool Print.) I call a procedure that unlocks all purgable segments except 1: the save segment for save or the print segment for Print.

This issue is only a small part of the larger issue of memory management in Mac programs: for example: should your grow zone function throw away undo information as a last ditch attempt to make extra room? How about if you need the room while in the middle of executing a ReDo command?

The simple answer is: don't print "max sizes handled" on your boxes. Have your program enforce conservative limits on the user. Maybe I'm just taking "the power to be your best" too much to heart.

You may have noticed that I answer more questions than I ask. Am I trying too hard on this issue?

> drs@bnlux0.bnl.gov (David R. Stampf)
--- David Phillip Oster -master of the ad hoc odd hack.

Summary/Conclusion: if your mac program calls a procedure in a segment that isn't loaded, the segment loader will call LoadSeg() to fetch it. If LoadSeg() fails, the mac crashes. This can bite you even if you never call UnloadSeg() since a segment only gets loaded when you call it for the first time (which might be AFTER you read a giant file.)

●●●

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: Re: **Handles and Virtual Memory (rewriting the Mem Mgr)**

In article <1669@intercon.com> amanda@mermaid.intercon.com (Amanda Walker) writes:

>Indeed, I've often been tempted to write a rudimentary version of the Mac
>memory manager for things like UNIX machines or PCs, simply so that I
>can have relocatable (and more importantly, *resizable*) blocks of memory
>without having to do my own memory management & reallocation.

Go for it! It will take you about 1 minute: consider:

```
typedef char *Ptr, **Handle;

Handle UnixNewHandle(size)long size;{
    Handle h;

    h = (Handle) malloc(sizeof(Handle));
    *h = (Ptr) malloc(size);
    return h;
}

void SetHandleSize(h, newSize)Handle h;long newSize;{
    realloc(*h, newsize);
}
```

does almost all the work for you. Error handling is left as an exercise. (The above assumes you have a virtual memory system, and a decent implementation of malloc(), so you don't have to worry about fragmentation of the heap.)

--- David Phillip Oster

●●●

From:siegel@endor.UUCP (Rich Siegel)
Subject: Re: Setting up multiple heaps

In article <21673@cup.portal.com> Andrew_James_Peterson@cup.portal.com writes:

>I'm going to try to set up an application that uses multiple heaps.

OK, but why?

Setting up a new heap zone is pretty simple:

```
        var
                hcZone: THz;

        procedure Heap;
                var
                        oldzone: THz;

        begin
                OldZone := GetZone;
                hcZone := THz(NewPtr(MaxZoneSize));
                InitZone(nil, 16, Ptr(Ord4(hcZone) + MaxZoneSize), Ptr(hcZone));
                SetZone(OldZone);
        end;
```

To make your new zone the current zone, do a

```
        SetZone(hcZone);
```

Multiple heaps generally aren't required, unless you're THINK Pascal or you want to perform some action on your own heap which requires blocks to be allocated, and you don't want to affect your own heap.

Rich Siegel

●●●

From:Earle R. Horton
Subject: Re: NewPtrclear,NewhandleClear

Matthew,

In article <3444@unix.SRI.COM> you write:
>I'm trying to get Offscreen sample source code from Apple to compile
>in LSP and its asking for Newptrclear and NewHandle clear. I think these are
>defined in MPW 3.0 but I can't afford to upgrade. Does someone have these
>definitions that they can send to me?

These are object dumps of part of the Interface.o library that comes with MPW 3.0, along with the C prototypes. They differ from the standard functions in that the trap word has an extra bit set so the system knows you want cleared space.

Earle R. Horton

```
dumpobj -h -m NEWPTRCLEAR {libraries}interface.o

Dump of file Boot:MPW:Libraries:Libraries:interface.o

Module:              Flags $08 Module="NEWPTRCLEAR"(309) Segment="Main"(273)
Content:             Flags $08
Contents offset $0000 size $000C
          MOVEA.L    (A7)+,A1
          MOVE.L     (A7)+,D0
          _NewPtr    ,Immed              ; A31E
          MOVE.L     A0,(A7)
          JMP        SAVERETA1           ; id: 287

dumpobj  -m SAVERETA1 {libraries}interface.o
Dump of file Boot:MPW:Libraries:Libraries:interface.o

; Note: SAVERETA1 starts here.  It stuffs the result code from D0
; into MemErr, then returns to the address stashed in A1.
```

```
0000000C: 2F09           '/.'              MOVE.L     A1,-(A7)
0000000E: 31C0 0220      '1.. '            MOVE.W     D0,$0220
00000012: 4E75           'Nu'              RTS

dumpobj -h -m NEWHANDLECLEAR {libraries}interface.o
Dump of file Boot:MPW:Libraries:Libraries:interface.o

Module:              Flags $08 Module="NEWHANDLECLEAR"(326) Segment="Main"(273)
Content:             Flags $08
Contents offset $0000 size $000C
          MOVEA.L    (A7)+,A1
          MOVE.L     (A7)+,D0
          _NewHandle  ,Immed              ; A322
          MOVE.L     A0,(A7)
          JMP        SAVERETA1           ; id: 287

pascal Handle NewHandleClear(Size byteCount);
pascal Ptr NewPtrClear(Size byteCount);
```

Note: These are declared as Pascal, and so exist in the object file, {libraries}interface.o, as uppercase symbols.

•••

From:siegel@endor.UUCP (Rich Siegel)
Subject:NewPtrclear,NewhandleClear

I wrote these inlines in 1987. (!!) Just change the names of the inlines, and you're in business.

```
FUNTION NewClearPtr (logicalSize : LongInt) : Ptr;
INLINE
        $201F, {move.l (a7)+, d0}
        $A31E, {_NewPtr, CLEAR}
        $2E88; {move.l a0, (a7)}
{Allocates a zeroed block of memory using _NewPtr, CLEAR}

FUNCTION NewSysPtr (logicalSize : LongInt) : Ptr;
INLINE
        $201F, {move.l (a7)+, d0}
        $A51E, {_NewPtr, SYS}
        $2E88; {move.l a0, (a7)}
{Allocates a block of memory in the system heap}

FUNCTION NewSysClearPtr (logicalSize : LongInt) : Ptr;
INLINE
        $201F, {move.l (a7)+, d0}
        $A71E, {_NewPtr, CLEAR+SYS}
        $2E88; {move.l a0, (a7)}

{Allocates a zeroed block in the system heap}

FUNCTION NewClearHandle (logicalSize : LongInt) : Handle;
INLINE
        $201F, {move.l (a7)+, d0}
        $A322, {_NewHandle, CLEAR}
        $2E88; {move.l a0, (a7)}
{Allocates a zeroed relocatable block}

FUNCTION NewSysHandle (logicalSize : LongInt) : Handle;
INLINE
        $201F, {move.l (a7)+, d0}
        $A522, {_NewHandle, SYS}
        $2E88; {move.l a0, (a7)}
{Allocates a relocatable block in the system heap}

FUNCTION NewSysClearHandle (logicalSize : LongInt) : Handle;
```

```
INLINE
        $201F, {move.l (a7)+, d0}
        $A722, {_NewHandle, SYS+CLEAR}
        $2E88; {move.l a0, (a7)}
{Allocates a zeroed relocatable block in the system heap}
```

•••

From: beard@ux1.lbl.gov (Patrick C Beard)
Subject: Re: StripAddress and pointer arithmetic
Summary: Only addresses which are master pointers.

In article <1990May25.194025.9751@csrd.uiuc.edu> bruner@sp15.csrd.uiuc.edu (John Bruner) writes:

```
#I've just read the technical note on StripAddress (#213, April 1990).
#On the second page under the (sub)heading "Ordered Address Comparison"
#it says
#
#       If you need to sort by address or do any other kind of
#       ordered address comparison, you need to call _StripAddress
#       on each address before doing any ordered comparisons
#       (>, <, >=, <=).  Remember, even though the CPU only uses
#       the lower 24 bits in 24-bit mode, it still uses all 32 bits
#       when performing arithmetic operations.
#
#Taken literally, this means that it is unsafe to write
#
#       struct something *p, things[64];
#
#       for (p = things; p < things + 64; p++)
#               ...
#
#because one can't rely upon the comparison "p < things+64" working.
```

Ok, time for a clarification. Note that your example implies variables that are allocated on the stack. This code will always work. Address arithmetic only has problems when you are dealing with addresses that might be dereferenced handles, or "master pointers". This is because the *current* memory manager keeps state information in the high byte (actually high nibble) of master pointers (i.e. if a handle is locked, purgeable, or a resource). These bits will throw off address comparisons obviously, and so StripAddress is in order. In general, addresses are clean. Examples of where they might not be include: dereferenced handles, code resources dereferenced then called will put garbage in the pc, and so should be stripped before being called.

A 32-bit operating system will make all this stuff go away. Let's hope we get it soon.
-
Patrick Beard, Macintosh Programmer (beard@lbl.gov) -

•••

From: shebanow@Apple.COM (Andrew Shebanow)
Subject: Re: StripAddress and pointer arithmetic

In article <1990May25.194025.9751@csrd.uiuc.edu> bruner@sp15.csrd.uiuc.edu (John Bruner) writes:

```
>Taken literally, this means that it is unsafe to write
>
>       struct something *p, things[64];
>
>       for (p = things; p < things + 64; p++)
>               ...
>
>because one can't rely upon the comparison "p < things+64" working.
```

I wrote that Tech Note, and I agree that it could be clearer (sigh).

The type of loop you show above will work fine without calling StripAddress. The only time you have to worry is if you are comparing two arbitrary (and possibly unrelated) addresses (for instance, if you compare a pointer parameter to a pointer stored in a list when doing a sort by address).

Sorry for the confusion,

Andrew Shebanow

•••

From: tecot@Apple.COM (Ed Tecot)
Subject: Re: IIci Trap dispatch Table

In article <4925@daffy.cs.wisc.edu> upl@gumby.cs.wisc.edu (Undergrad Projects Lab) writes:

> >I am working on a program which depends on the location of the IIci Trap
> >Dispatch Tables. The problem is, they don't seem to be in the place described
> >in IM IV (page 13). I have been able to find the OS Traps, but I need to know
> >where the Toolbox Traps are. Is there any one out there (APPLE?) who can give
> >me that info?

"I can tell you, but I'll then I'll have to kill you."

Bad news. The trap dispatch tables are not guaranteed to be in a certain location. They've been forced to move with each new ROM and system as the size and format of the table grows. You could probably reverse-engineer the information for your particular ROM and system, but I can guarantee that the location will change on other machines and in the future.

"There is another..."

Use NGetTrapAddress to access the dispatch table. It always knows where to find it, even if it's split into several pieces.

"Live long and prosper."

I hope I've helped you.

_emt

•••

From: lim@iris.ucdavis.edu (Lloyd Lim)
Subject: Identifying real handles (with Code)

This is perhaps not the best thing to do on a Mac but I need to be able to tell if some arbitrary long is a real handle in the System heap or the current app heap. I want to keep it relatively clean so I don't want to go looking through the heap's internal structures. Currently, I use the following code:

```
Boolean ValidHandle(address)

register long address;

{
   register Boolean  valid;
   register THz      heapZone;

   valid = FALSE;
   if (address && !(address & 1)) {
      heapZone = HandleZone(address);
      if (!MemError() && (heapZone == SystemZone() ||
                          heapZone == ApplicZone())) {
         valid = TRUE;
      }
   }
   return(valid);
```

```
}
```

The problem is that this routine can get passed any arbitrary long and that some values seem to cause a bus error with HandleZone. I say "seem" because I haven't been able to find a value which causes a bus error in a test program but bus errors do occur in the real situation if enough values are examined. The real situation is practically unobservable.

Any ideas on a better way? If you try doing extra checking before calling HandleZone, you'll get bus errors right away when you do (*(Ptr) address). If there is a solution that isn't right 100% of the time but doesn't cause bus errors, that would also be ok.

+++
Lloyd Lim Internet: lim@iris.ucdavis.edu (128.120.57.20)

●●●

From: russotto@eng.umd.edu (Matthew T. Russotto)
Subject: Re: Identifying real handles

In article <7426@ucdavis.ucdavis.edu> lim@iris.ucdavis.edu (Lloyd Lim) writes:
> >This is perhaps not the best thing to do on a Mac but I need to be able to
> >tell if some arbitrary long is a real handle in the System heap or the current
> >app heap. I want to keep it relatively clean so I don't want to go looking
> >through the heap's internal structures. Currently, I use the following code:

Check to see if the handle itself is below BufPtr or MemTop or whatever variable you feel is appropriate. If it's below that, there should never be a bus error. Then do validity checks on *address, and check to see if IT is below BufPtr or MemTop. Then call HandleZone.

--
Matthew T. Russotto russotto@eng.umd.edu russotto@wam.umd.edu

●●●

From: ted@cs.utexas.edu (Ted Woodward)
Subject: Re: Need large array on Think C.

In article <265986A9.26645@paris.ics.uci.edu> jchoi@paris.ics.uci.edu (John Choi) writes:

> > Sorry for this trival question, but I really need to get this done.

no prob; that's what this group is here for...

> >When I declare an array ('char arr[100][300]'), the complier
> >gives me an 'illegal array bounds' error. Is there any way I can
> >increase this limit?

You are limited to 32K of global and static variables because of the compiler. This is because the 68000 can only have a 16 bit offset to an address reg, the technique used for global vars. But there is a way. Read on...

> > The program uses array notation throughout and I don't want to
> >recode using pointers and malloc(). I just need to increase the array
> >size. Is there a complier option I need to set using Think C 4.0

No compiler option. You have to use pointers. But, because of the way C does arrays and pointers, you can address a pointer as an array. Because you have a multidimensional array, C needs to know how big each dimension is. Actually, C only needs to know how big the 1st dimension is, and doesn't care how big the second is (same as when you declare an array).

Try this:

```
#define MAXX 100
#define MAXY 300
```

```
char (*arr)[MAXX-1]  /* C arrays go from 0-n */
       /* this declares a pointer to an array of MAXX chars */

main()
{   arr = (char *) NewPtr(sizeof(char) * (long) MAXX * MAXY);
etc...
}
```

this will give you the array, addressable like a normal array. You need to declare it like above so it knows how many elements in the 1st parameter, and the NewPtr actually allocates the space on the heap. You could (and possibly should) use a handle, but this makes it hard to access in this case. Instead, do this FIRST so you don't frag the heap. You don't really need the sizeof(char) because that is 1 byte for char, but you do need the type coercion to long there because NewPtr wants a long, and because you might overflow an int.

> Thanks for the help.

No prob. Hope this solves the problem...

--
Ted Woodward (ted@cs.utexas.edu)

●●●

From: bruner@sp15.csrd.uiuc.edu (John Bruner)
Subject: Re: Need large array on Think C.

In article <736@grit.cs.utexas.edu>, ted@cs (Ted Woodward) writes:
]In article <265986A9.26645@paris.ics.uci.edu> jchoi@paris.ics.uci.edu (John Choi) writes:
]>When I declare an array ('char arr[100][300]'), the complier
]>gives me an 'illegal array bounds' error. Is there any way I can
]>increase this limit?
]
]Try this:
]
]#define MAXX 100
]#define MAXY 300
]
]char (*arr)[MAXX-1] /* C arrays go from 0-n */
]   /* this declares a pointer to an array of MAXX chars */
]
]main()
]{   arr = (char *) NewPtr(sizeof(char) * (long) MAXX * MAXY);
]etc...
]}

Ted is close, but not quite correct. You need to declare the pointer to the array using

```
char (*arr)[MAXY];
```

This technique works if the dimensions of the array (except the most-significant one) are constants at compile time; otherwise, you need to use arrays of pointers and multiple levels of indirection (using "arr[x][y]" notation) or explicit index calculation.

John Bruner Center for Supercomputing R&D, University of Illinois

●●●

From: stoms@castor.ncgia.ucsb.edu (David Stoms)
Subject: Re: Need large array on Think C. (explanation on how to do it)

In article <265986A9.26645@paris.ics.uci.edu> jchoi@paris.ics.uci.edu (John Choi) writes:
> Sorry for this trival question, but I really need to get this
>done. When I declare an array ('char arr[100][300]'), the complier
>gives me an 'illegal array bounds' error. Is there any way I can

>increase this limit?

I'm getting tired of this question so I think I'll try to give a generic answer to cover any further problems.

When you want lots of variable space >32K in Think C or any compiler that limits globals to 32K, you have to use the Memory Manager (gasp!). If you want a big array, such as "char arr[100][400]" then you need to sever your dependancy from the compiler and get your hands dirty.

To allocate this array:

```
char    *arr;

arr = NewPtr((Size)100*400);
```

Then to use the array you can index like this:

```
x = 20; y = 32;
*(arr+x+y*100) = 'H';
```

Thats it! If you want to use a handle the only big difference is:

```
*(*arr+x+y*100)  << note the extra * >>
```

If you do this, be sure to lock down the handle when your r.h.s. could move memory.

Josh.

●●●

From: dudek@ai.toronto.edu (Gregory Dudek)
Subject: Re: Need large array on Think C.

In article <5506@hub.ucsb.edu> stoms@castor.ncgia.ucsb.edu () writes:
>In article <265986A9.26645@paris.ics.uci.edu> jchoi@paris.ics.uci.edu (John Choi) writes:
>> Sorry for this trival question, but I really need to get this
>>done. When I declare an array ('char arr[100][300]'), the complier
>>gives me an 'illegal array bounds' error. Is there any way I can
>>increase this limit?
>
>
>When you want lots of variable space >32K in Think C or any compiler
>that limits globals to 32K, you have to use the Memory Manager (gasp!).
>If you want a big array, such as "char arr[100][400]" then you need to
>sever your dependancy from the compiler and get your hands dirty.
>
>To allocate this array:
>
>char   *arr;
>
>arr = NewPtr((Size)100*400);
>
>Then to use the array you can index like this:
>x = 20; y = 32;
>*(arr+x+y*100) = 'H';
>
>Thats it! If you want to use a handle the only big difference is:
>*(*arr+x+y*100) << note the extra * >>
>
>If you do this, be sure to lock down the handle when your r.h.s. could

Although the method above works fine, I find it a bit ugly. Not only does sticking in calls to NewPtr explicitly lose portability, but the indexing scheme is much less readable than regular arrays. Finally, you may want to have the arrays initialized to zero, like ``real'' ones.

I prefer the following scheme which is much more isomorphic to the commonly used sytax. Note, also, that since the arrays are allocated using pointers not handles, no locking down of memory is required at any time.

1) Add this header code:
==========

```
#define DCLARRAY(type,name,y,x)     type *name[y]
#define INITARRAY(type,name,y,x)    allocate(name,(int)sizeof(type),y,x)


allocate(array,elementsize,y,x)
char *array[];
int elementsize;
int y,x;
{
    register int ix, iy;
    for (iy=0;iy<y;iy++) {
    array[iy] = NewPtr((long)(x*(long)elementsize));
    if (!array[iy]) exit(1);
    }
    /* init array to zero here, if desired */
}
```

2) Replace array declarations of the form:
```
  float foo[999][566]       with
  DCLARRAY(float,foo,999,566)
```

3) Insert this before the array would be used:
```
    INITARRAY(float,foo,999,566)
```

4) Use the array like normal, i.e. foo[23][21]

Greg Dudek

●●●

From: odawa@well.sf.ca.us (Michael Odawa)
Subject: Re: Identifying real handles

In article <7426@ucdavis.ucdavis.edu> lim@iris.ucdavis.edu (Lloyd Lim) writes:

> ...I need to be able to tell if some arbitrary long is a real handle in
> the System heap or the current app heap. I want to keep it relatively clean
> so I don't want to go looking through the heap's internal structures...The
> problem is that this routine can get passed any arbitrary long and that some
> values seem to cause a bus error with HandleZone....Any ideas on a better
> way?

You have to make a range check against the high end of your memory. Here's something you might add to your code:

```
{
   register Boolean  valid;
   register THz      heapZone;

   valid = FALSE;
   if (address && !(address & 1)) {
      if (address < long(**ApplicZone.BkLim)) {    /***** Add this line *****/
          heapZone = HandleZone(address);
          if (!MemError() && (heapZone == SystemZone() ||
                              heapZone == ApplicZone())) {
            valid = TRUE;
```

```
      }
   }
   return(valid);
}
```

A different check (which I do in my version of ValidHandle) might be to determine whether the tag byte on the block header (the first of the eight bytes which preceed the address) contained 0x8x, as documented in IM II-24:

```
    if (address < long(**ApplicZone.BkLim))
        if ((*(ptr)(address - 8) & 0x80) != 0)
            valid = TRUE;
```

Of course, we have not even begun to discuss problems with 24-bit vs 32-bit addressing schemes. So perhaps we ought to preceed our code by

```
    address = (long)StripAddress((ptr)address);
```

Michael Odawa

●●●

Chapter 9 Multi-Finder Madness

Context switching under MultiFinder.....132
Inhibiting major switching under MultiFinder.....132

From: mc@fctunl.rccn.pt (Miguel Calejo)
Subject: Re: Context switching under MultiFinder

In article <1990May16.212252.16796@cec1.wustl.edu> wilcox@wucs1.wustl.edu (Charles D Wilcox) writes:

> > Just wanted to thank everyone who has answered my stupid :-) question about
> > context switching under MultiFinder. The answer, which would be obvious to
> > anyone who took the time to think, is to call OpenDeskAcc() with the program
> > name. Next time, I promise to think harder (and smarter).

Some experiences I did led me to prefer using Get/WaitNextEvent loops instead, using a common flag to control things. This allows activation of background applications, AND I found it to be slightly faster than OpenDeskAcc(). By the way, on a Mac II all I got were 12 context switchings per second (6 "remote procedure calls").

Has someone made experiences with System 7.0 alpha ? Since applications seem to continue without private memory space, is the context switching improved or not ?

Miguel Calejo

●●●

From: dm@everexn.uucp (Dan McMullen)
Subject: Re: Inhibiting major switching under MultiFinder

In <1990Jun10.010643.5803@cs.UAlberta.CA> news@cs.UAlberta.CA (News Administrator) writes:

> >Is there a way I can get MultiFinder to suspend
> >major switching at this point without going back to a dBoxProc dialog,
> >or putting image display support into my own program?

Currently, multifinder will refuse to do a major task switch if:

A. The varCode of the window is 1 (==dBoxProc), OR
B. The high bit of the spareFlag in the windowRecord is set.

This info from a recent query i made to macdts. I got bit by this behavior when a custom WDEF used varCode==1, and would refuse to do the task switch like I expected, so that's one possible solution. Setting the bit in spareFlag might work, but I haven't tried this.

Any comments from Apple as to whether these solutions will continue to work in the future?

--
Dan McMullen
●●●

Chapter 10 Printing & the Print Manager

Choose printer from software?.....134
Generic Printing code (complete Source).....134

From: mm5l+@andrew.cmu.edu (Matthew Mashyna)
Subject: Re: Choose printer from software?

Charles Oldham writes:
> Is there a way to choose a printer through software? I.e.
> not using the Chooser ?

I was trying to do this and did get it to work. I got flamed a lot for asking how to do it too!

<flame guard on>

You can set the printer selection by reading in the PAPA resource -8192 from the LaserWriter cdev.The resource contains the pascal strings for the name, type and zone followed by the net address -- net, node & socket (order ?). If my memory's right it's 103 bytes (better check.)

The crudest thing you can do is zero out the resource and stuff your string values for the name type and zone into the resource. The first time you goe to print the LaserWriter drivers will have to work a little harder to find the propper address (because you stuffed in 0's). You could go farther and do an NBP lookup for the address and finish the job too.

I know this is really a bad thing to do but I needed to do it for a TCP/IP LPR deamon to direct output from Unix & VMS machines to many Appletalk printers.

Matt Mashyna

●●●

From: Richard M. Siegel
Subject: Generic Printing code (complete Source)

[Rich posted this to the net a while back. MXM]

```
{UGenericPrint.p}
{©1989 Richard M. Siegel, all rights reserved}


{UGenericPrint provides a standard interface for programs that wish to print images}
{using the Print Manager.}
{It will take care of most of the gory details of printing.}

{History:}
{}
{   4/12/89    RMS    Creation}

{Lightspeed Pascal source options: Courier 10, 4 spaces per indent, 4 spaces per tab.}

unit UGenericPrint;
interface

{For MPW Pascal, the USES list will have to be changed. Also, you may wish to use MacPrint
if}
{you're going to run on versions of the System earlier than 4.1}
uses
    MacPrint, editorGlobals;

{Function "GenericPrint" handles all of the printing details. Its arguments
    }
{are:
                  }
{
                  }
{   hPrint :   A Print Manager handle. If hPrint is NIL, then GenericPrint
    }
```

```
{               will allocate its own default print handle and dispose it
    }
{               before exiting.
                    }
{
                    }
{   statusDialog :    A pointer to a dialog box which will be displayed while
    }
{                   printing. If statusDialog is NIL, no dialog will be displayed.      }
{
                    }
{
                    }
{   UserPrepProc :    A function that performs any pre-printing initializations that   }
{                   may  be required. It should return TRUE if it completed
    }
{                   successfully, FALSE to abort printing. One thing that the           }
{                   UserPrepProc MUST DO: it is imperative that the UserPrepProc set}
{                   hPrint^^.prJob.iLstPage to the correct number of pages in the       }
{                   document. If all pages are specified in the print dialog, then      }
{                   this field of the print handle will be 9999; if this is the case,}
{                   set it to the correct number of pages.
        }
{
                    }
{   UserPageProc :    A function that does the actual drawing on the page.  It is
    }
{                   supplied with a print handle, a rectangle representing the          }
{                   drawing area, and a page number. UserPageProc should return         }
{                   TRUE if it completed successfully, or FALSE if you wish to          }
{                   abort the printing process. The current grafPort will be the }
{                   printing grafPort, so if your routine changes it, be sure to }
{                   restore it.
                        }
{
                    }
{   GenericPrint will return noErr if printing was completed successfully,
    }
{   and a negative OS result code otherwise.
                }

function GenericPrint (hPrint: THPrint; statusDialog: DialogPtr; function UserPrepProc
(hPrint: THPrint): Boolean; function UserPageProc (hPrint: THPrint; pageRect: Rect; pageNum:
Integer): Boolean): OSErr;

implementation
function GenericPrint (hPrint: THPrint; statusDialog: DialogPtr; function UserPrepProc
(hPrint: THPrint): Boolean; function UserPageProc (hPrint: THPrint; pageRect: Rect; pageNum:
Integer): Boolean): OSErr;
    type
        TPrDevice = (MacScreen, ImageWriter, Unknown, LaserWriter);

    var
        scratch: Boolean;

        wasNIL: Boolean;      {TRUE if we needed to allocate hPrint}

        curPrError: OSErr;           {The last printing error}
        PrIsOpen: Boolean;           {TRUE if we're in the middle of a PrOpen/PrClose pair}
        DocIsOpen: Boolean;          {TRUE if we're in a PrOpenDoc/PrCloseDoc pair}
        PageIsOpen: Boolean;  {TRUE if we're in a PrOpenPage/PrClosePage pair}

        nCopies: Integer;     {Number of copies to be printed.}
        prDevice: TPrDevice;  {The kind of printer we're printing on}
        draftMode: Boolean;          {Draft or spool?}
```

```
        prPort: TPPrPort;
        prStatus: TPrStatus;

        savePort: GrafPtr;

        i, p: Integer;

    procedure CleanUp; {This procedure cleans up after the routine and exits.}
        begin
            if wasNil and (hPrint <> nil) then
                DisposHandle(Handle(hPrint));

            if PageIsOpen then
                PrClosePage(prPort);

            if DocIsOpen then
                PrCloseDoc(prPort);

            if PrIsOpen then
                PrClose;

            SetPort(savePort);

            GenericPrint := curPrError;
            Exit(GenericPrint);
        end;

    procedure CheckPrError;
        begin
            if curPrError <> noErr then
                CleanUp;
        end;

    begin
        GenericPrint := noErr;
        PrIsOpen := False;
        DocIsOpen := False;
        PageIsOpen := False;

        GetPort(savePort);

        wasNil := (hPrint = nil);

        PrOpen;
        curPrError := PrError;
        CheckPrError;
        PrIsOpen := True;

{If we need to, allocate the print handle.  If the allocate failed, exit out}
{with a memFulErr result.}
        if wasNil then
            begin
                hPrint := THPrint(NewHandle(SizeOf(TPrint)));
                if hPrint = nil then
                    begin
                        curPrError := memFullErr;
                        CleanUp;
                    end;
            {Initialize the print handle}
                PrintDefault(hPrint);
            end;

{Validate the print handle.}
        scratch := PrValidate(hPrint);

        if not PrJobDialog(hPrint) then
```

```
            CleanUp;

{Call the user's prep proc, and exit if it returns false.}
        if not UserPrepProc(hPrint) then
            CleanUp;

{We need to determine some parameters to print correctly.}

        {Figure out which printer we're using}
        prDevice := TPrDevice(BSR(hPrint^^.prStl.wDev, 8));

        {Determine whether we're in draft mode or not.}
        draftMode := not Odd(hPrint^^.prJob.bJDocLoop);

        {If we're in draft mode on an ImageWriter, we have to honor the number of}
        {copies; otherwise, the printer driver takes care of it for us.}
        if draftMode and (prDevice = ImageWriter) then
            nCopies := hPrint^^.prJob.iCopies
        else
            nCopies := 1;

{Now we're ready to begin the printing loop.}

        {Open the print document. If it fails, leave.}
        prPort := PrOpenDoc(hPrint, nil, nil);
        docIsOpen := True;
        curPrError := PrError;
        CheckPrError;

        SetPort(@prPort^.gPort);
        for i := 1 to nCopies do
            begin
                for p := hPrint^^.prJob.iFstPage to hPrint^^.prJob.iLstPage do
                    begin
                        thePage := p;
                        PrOpenPage(prPort, nil);
                        PageIsOpen := True;
                        scratch := UserPageProc(hPrint, hPrint^^.prInfo.rPage, p);
                        PrClosePage(prPort);
                        if not scratch then
                            begin
                                curPrError := iPrAbort;
                                CleanUp;
                            end;
                    end;
            end;

        PrCloseDoc(prPort);
        DocIsOpen := False;
        curPrError := PrError;

        if (not DraftMode) and (curPrError = noErr) then
            begin
                PrPicFile(hPrint, nil, nil, nil, prStatus);
                curPrError := PrError;
            end;
    end;
end.
```

Chapter 11 QuickDraw™ & Graphics

Copybits and the colortable 140
Piccomment and technote 175 140
Sync Drawing (with Code) 141
THINK C & 32-bit quickdraw (interfaces for 32bit quickdraw)..... 143
THINK C & 32-bit quickdraw (.c) (interfaces for 32bit quickdraw) 145
How do I get the slot of the main screen? 149
Drawing to an off screen bitmap (with code) 150
Color Quickdraw and copybits glitches
(ps to colorqQD with Code) 151
Finding the size of the screen under Multifinder/Color QD
(gdevices) 156
Bitmap not showing up in scrapbook (cliprect) 157
Finding the size of the screen under Multifinder/Color QD 157,158
Marquee Help? (with Code)..... 158
How to get the QD globals outside of an application 162
Marquee Help? 163
Finding the size of the screen under Multifinder/Color QD 163,164
PICT resource to BitMap (with code)..... 165
Writing large PICT files??? 166
How's the screen cleaned after _SysError? 166
Optimizing Copybits (good Information with code) 167
Fast Copybits suggestions needed, other questions 169
How does one set up an offscreen buffer 169
Rotate Bit Map CCW (full source code)..... 171
Aligning bitmaps? (with code)..... 182,184
ThePort and TENew (problem with font) 184
FatBits,thinbits and MapPt() 185
32-bit QuickDraw scaling behavior 185
Need help creating a Quickdraw picture..... 186
Still need help creating a QuickDraw picture 187
Need help animating with 24 bit card..... 187
Setting up CLUTs - Advise??????? 188
Bad pixmap data written by OpenPicture & CopyBits..... 188
Fading ...(dissolve effect)..... 189
Drawing all over the desktop 189
PICT hacking question 190
Wanted- a window that comes up in black 190,191
Dragging Bitmaps? 191
Wanted- a window that comes up in black 192
Dragging Bitmaps? 192
2-16-256-million color mode 192
Reading PICT's revisited. 193
How to read a Pict into a scrollable window 197

From: Larry Rosenstein, Apple Computer, Inc.
Subject:Copybits and the colortable

In article <3084@ur-cc.UUCP> tonyg@merlin.cvs.rochester.edu (Tony Giaccone) writes:

> So if that's the case why doesn't the copybits install a copy of the
> current color table into the PICT2. Or, if it does how come PICTViewer
> doesn't use that Color Table when it displays the file? It certainly seems
> to use the color tables in other PICT2 files. I don't get it.

If you were to dump the picture, I think you would find that your custom color table was present. A pixmap always has an associated color table.

What happens is that the CopyBits from your picture to the screen follows the normal Color QuickDraw rules. That is, Color QuickDraw maps the desired colors to the closest ones that the display can show. Color QuickDraw never modifies the current color table in the process of drawing. If you were using a 24-bit display, then you would get the desired colors, because all colors are achievable.

This is a known weakness in the PICT format. What programs needs to do is get the "appropriate" color table for the picture, set up that palette, and then draw the picture. The problem is that there is no standard for getting the "appropriate" color table.

Some programs store the CLUT as a separate resource, and read that resource to construct the palette. (You could use the resource type understood by the Palette Manager.)

Another approach is to scan through the picture and build up a CLUT based on the colors actually used. For a bitmap image this is OK, since the CLUT is stored in the pixmap. For an arbitrary picture it is not as easy to do.

-
Larry Rosenstein, Apple Computer, Inc.,Object Specialist

•••

From: Tom Dowdy
Subject: Piccomment and technote 175

I've watched lots of interesting postings on peoples theories on how to make this work, but I haven't yet seen what I consider the best answer.

In article <UZB_yoW00UzxQ1R4wY@andrew.cmu.edu> ba0k+@andrew.cmu.edu (Brian Patrick Arnold) writes:

> [Recommends use of :]
> "SetLineWidth" described in Tech Note #175.

This is a correct way to get thin lines from a LaserWriter. Note that this PicComment is only currently interpreted on the LaserWriter and you gain nothing on other printers.

BUT:
There is a way to get thin lines and device positioning from *all* printer drivers. This is to use the PrGeneral calls GetRslData and SetRsl, as documented in Tech Note 128. When you make these calls, your app is said to be "imaging at device resolution."

This means that within your print record you will see the page expressed in terms of the resolution that you specified. This means that you can then place/draw objects within the coordinate space of the device, rather than the coordinate space of 72 dpi, which is what normally happens.

The reason this approach is better is that owners of LaserWriter SCs, DeskWriters (and this is a growing number), ImageWriter LQs, and even lowly ImageWriters will be able to draw things to best possible output for their device. Everyone is happy! And you didn't have to write any PostScript, or mess with the Enlarge/Reduce box in the dialog, or even use PicComments.

Please note that you do not need to use these calls if you application is simply trying to get "smooth looking circles" from the LaserWriter. The drivers take care of that for you. You only need to make this call if you need to draw lines at device resolution, or you need to be able to place dots or objects within the coordinate space of the device.

If you'd like to make these calls, please refer to the Tech Note for information on when/how to call GetRslData and SetRsl. Those of you with Volume V of Inside Mac can also find the information there in the chapter on the Printing Manager, the information looks the same, although in general I tend check the Tech Notes first because they are updated more often.

Tom Dowdy Internet: dowdy@apple.COM

•••

From:Dave ?
Subject:Sync Drawing (with Code)

To do synched drawing on a Mac II, you want to use a technique analogous to that of watching TickCount() on the MacPlus, but you have to make your own VBL counter. This should be done with VBLTask tied to the video slot so you really get video synching.

To illustrate this, here is some code (in Think C):

Dave

```
/*  my globals  */
extern long     VBLcounter;
extern short    videoSlot;
extern VBLTask  drawtask;

/*************************************************************

    Before we do anything, we have to know which slot the
    card is in.  Use this routine.  (See Start Manager in
    IM V.)

*/

FindVideoSlot()
{
    DefVideoRec defvidrec;

    GetVideoDefault(&defvidrec);
    videoSlot = defvidrec.sdSlot;
}

/*********************************************************************

    Then we have to install a VBL Task.  This is the one
    I'll use.  To use global variables you must setup the
    A5 register.  To do this in MultiFinder requires
    a special trick, which is demonstrated with the
    CacheA5 function.

*/

pascal void ScreenDrawTask()
{
/*
*   WARNING! - this is self-modifying code, but it is
*   necessary to run under MultiFinder...see Tech Note 180
*/
    long    oldA5;

    asm {
        move.l  a5,oldA5
        move.l  #0,a5   ; this will be replaced with
                        ; CurrentA5 by the CacheA5 function at runtime
    }
    VBLcounter += 1;
    drawtask.vblCount = 1;
```

```
    asm {
        move.l  oldA5,a5
    }
}

pascal void CacheA5(addr)
long    addr;
/*
*   This function will place CurrentA5 into the beginning of task
*   functions which start as:
*   pascal void FooTask()
*   {
*       long    junk;
*
*       asm {
*           move.l  a5,junk
*           move.l  #0,a5   ;this will be replaced by CurrentA5
*           ....
*
*   See Tech Note #180
*/
{
        asm {
            move.l  addr,a0
            add.w   #2,a0   ; move past JMP instruction
            move.l  (a0),a0
            add.w   #0xa,a0 ; move past first part of function
            move.l  CurrentA5,(a0)
                            ; puts CurrentA5 in (*addr)() dc.l location
    }
}


/*************************************************************

    This function actually sets up the VBL Task.

*/

StartCounter()
{
    CacheA5(ScreenDrawTask);
    VBLcounter = 0;
    drawtask.vblAddr = ScreenDrawTask;
    drawtask.qType = vType;
    drawtask.vblCount = 1;
    drawtask.vblPhase = 0;
    SlotVInstall(&drawtask,videoSlot);
}


/*************************************************************

    This function removes the VBL Task.

*/

RemoveCounter()
{
    SlotVRemove(&drawtask,videoSlot);
}


/*************************************************************

    This function holds everything until it's
```

```
    time to draw. Note that it could be replaced by
    a #define macro (make VBLjunk a global).

*/

WAIT_FOR_SCREEN()
{
    long VBLjunk;

    VBLjunk = VBLcounter;
    do {} while ( VBLcounter == VBLjunk );
    /* this is used to synch drawing to screen */
}

/*****************************************************************

    Now, putting this altogether into a program

*/


main()
{

    /* do your init stuff */

    FindVideoSlot();
    StartCounter();

    /* now, whenever you want to draw something, do this... */
    WAIT_FOR_SCREEN();
    DrawMyThing();

    /* when you're done...*/

    RemoveCounter();

}
```

•••

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: THINK C & 32-bit quickdraw (interfaces for 32bit quickdraw)

As my way of saying thank you, here are the interfaces for 32-bit Quickdraw for THINK C. You may want to save this file and upload it,since it contains tabs and fairly long lines.:

--- David Phillip Oster -master of the ad hoc odd hack.

```
(cut here, call this QuickDraw32Bit.h).
/* let's get a real juicy 32 bit discussion going based on this maybe !!
   by bhamlin@well.sf.ca.us (Brian M. Hamlin)
   from oster@well.sf.ca.us (David Phillip Oster)
 */

/* I think that's it - check it out, write something fun. Remember: You
       don't have to have a 32bit video card, just Sys 6.0.3 or greater on
       a color-capable machine and the 32bit Quickdraw INIT in your system
       folder to use these calls !
 */
/* hoo-boy, well ANYWAY thanks to Sam Roberts for assembler tweaking. */


#ifndef _QuickDraw32Bit_
#define _QuickDraw32Bit_
```

```
#ifndef _Quickdraw_
#include <Quickdraw.h>
#endif

#ifndef _Color_
#include <Color.h>
#endif

/* New Constants for 32-Bit QuickDraw */

#define ditherCopy 64               /* Dither mode for Copybits */
#define RGBDirect 16         /* 16 & 32 bits/pixel pixelType value */


/* New error codes */

#define rgnOverflowErr -147                /* Region accumulation failed. Resulting region
may be currupt */
#define pixmapTooDeepErr -148              /* Pixmap is not 1-bit/pixel for BitmapToRegion
*/
#define insufficientStackErr -149   /* QuickDraw could not complete the operation */
#define cDepthErr -157                              /* invalid pixel depth passed to
NewGWorld or UpdateGWorld */


#define       pixPurgeMask          1
#define       noNewDeviceMask              2

/* Flag bits passed to or returned by Offscreen routines */

enum  {
       pixPurgeBit = 0,
       nowNewDeviceBit = 1,
       pixelsPurgeableBit = 6,
       pixelsLockedBit = 7,

       mapPixBit = 16,                     /* set if color table mapping occurred */
       newDepthBit = 17,           /* set if pixels were scaled to a different depth */
       alignPixBit = 18,           /* set if pixels were realigned to screen alignment */
       newRowBytesBit = 19,  /* set if pixmap was reconfigured in a new rowBytes */
       reallocPixBit = 20,         /* set if offscreen buffer had to be reallocated */
       clipPixBit = 28,            /* set if pixels were or are to be clipped */
       stretchPixBit = 29,         /* set if pixels were or are to be stretched/shrinked */
       ditherPixBit = 30,
       gwFlagErrBit = 31
};

typedef enum {
       pixPurge = 1L << pixPurgeBit,
       nowNewDevice = 1L << nowNewDeviceBit,
       pixelsPurgeable = 1L << pixelsPurgeableBit,
       pixelsLocked = 1L << pixelsLockedBit,
       mapPix = 1L << mapPixBit,
       newDepth = 1L << newDepthBit,
       alignPix = 1L << alignPixBit,
       newRowBytes = 1L << newRowBytesBit,
       reallocPix = 1L << reallocPixBit,
       clipPix = 1L << clipPixBit,
       stretchPix = 1L << stretchPixBit,
       ditherPix = 1L << ditherPixBit,
       gwFlagErr = 1L << gwFlagErrBit
}GWorldFlag;

typedef long GWorldFlags;

/* Type definition of a GWorldPtr */
```

```
typedef CGrafPtr GWorldPtr;


/* Function Prototypes (necessary to get automatic type coercion -- see LSC User's Manual
p.125)
 */
pascal OSErr   BitmapToRegion(RgnHandle, BitMap *);
pascal QDErr   NewGWorld(GWorldPtr *, short, Rect *, CTabHandle, GDHandle, GWorldFlags);
pascal Boolean        LockPixels(PixMapHandle);
pascal void           UnlockPixels(PixMapHandle);
pascal GWorldFlags    UpdateGWorld(GWorldPtr *, short, Rect *, CTabHandle, GDHandle,
GWorldFlags);
pascal void           DisposeGWorld(GWorldPtr);
pascal void           GetGWorld(CGrafPtr *, GDHandle *);
pascal void           SetGWorld(CGrafPtr, GDHandle);
pascal void           CTabChanged(CTabHandle);
pascal void           PixPatChanged(PixPatHandle);
pascal void           PortChanged(GrafPtr);
pascal void           GDeviceChanged(GDHandle);
pascal void           AllowPurgePixels(PixMapHandle);
pascal void           NoPurgePixels(PixMapHandle);
pascal GWorldFlags    GetPixelsState(PixMapHandle);
pascal void           SetPixelsState(PixMapHandle, GWorldFlags);
pascal Ptr            GetPixBaseAddr(PixMapHandle);
pascal QDErr   NewScreenBuffer(Rect *, Boolean, GDHandle *, PixMapHandle *);
pascal void           DisposeScreenBuffer(PixMapHandle);
pascal GDHandle       GetGWorldDevice(GWorldPtr);

#endif
```

●●●

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject:Re: THINK C & 32-bit quickdraw (.c) (interfaces for 32bit quickdraw)

As my way of saying thank you, here are the interfaces for 32-bit Quickdraw for THINK C. You can save this, compile it, then get at all the 32-bit quickdraw traps from THINK C. You may want to save this file and upload it, since it contains tabs and fairly long lines.:

>--- David Phillip Oster -master of the ad hoc odd hack.

```
(cut here, call this QuickDraw32Bit.c).
#include <QuickDraw32Bit.h>

/* let's get a real juicy 32 bit discussion going based on this maybe !!
   by bhamlin@well.sf.ca.us (Brian M. Hamlin)
   from oster@well.sf.ca.us (David Phillip Oster)
 */

/* BitmapToRegion -
 */
pascal OSErr BitmapToRegion(region, bmap)
       RgnHandle      region;
       BitMap         *bmap;{

       asm {
              SUBQ.L #2,SP
              MOVE.L region(A6),-(SP)
              MOVE.L bmap(A6),-(SP)
              DC.W   0xA8D7
              MOVE.W (SP)+,D0
       }
}

/* NewGWorld -
```

```
 */
pascal QDErr NewGWorld(offscreenGWorld, pixelDepth, boundsRect, cTable, aGDevice, flags)
       GWorldPtr             *offscreenGWorld;
       short                 pixelDepth;
       Rect                  *boundsRect;
       CTabHandle            cTable;
       GDHandle              aGDevice;
       GWorldFlags           flags;{

       asm {
              SUBQ.L #2,SP
              MOVE.L offscreenGWorld(A6),-(SP)
              MOVE.W pixelDepth(A6),-(SP)
              MOVE.L boundsRect(A6),-(SP)
              MOVE.L cTable(A6),-(SP)
              MOVE.L aGDevice(A6),-(SP)
              MOVE.L flags(A6),-(SP)
              MOVEQ  #0,D0
              DC.W   0xAB1D
              MOVE.W (SP)+,D0
       }
}

/* LockPixels -
 */
pascal Boolean LockPixels(pm)PixMapHandle  pm;{
       asm {
              SUBQ.L #2,SP
              MOVE.L pm(A6),-(SP)
              MOVEQ  #1,D0
              DC.W   0xAB1D
              MOVE.B (SP)+,D0
       }
}

/* UnlockPixels -
 */
pascal void UnlockPixels(pm)PixMapHandle   pm;{
       asm {
              MOVE.L pm(A6),-(SP)
              MOVEQ  #2,D0
              DC.W   0xAB1D
       }
}

/* UpdateGWorld -
 */
pascal GWorldFlags UpdateGWorld(offscreenGWorld, pixelDepth, boundsRect, cTable, aGDevice,
flags)
       GWorldPtr             *offscreenGWorld;
       short                 pixelDepth;
       Rect                  *boundsRect;
       CTabHandle            cTable;
       GDHandle              aGDevice;
       GWorldFlags           flags;{

       asm {
              SUBQ.L #4,SP
              MOVE.L offscreenGWorld(A6),-(SP)
              MOVE.W pixelDepth(A6),-(SP)
              MOVE.L boundsRect(A6),-(SP)
              MOVE.L cTable(A6),-(SP)
              MOVE.L aGDevice(A6),-(SP)
              MOVE.L flags(A6),-(SP)
              MOVEQ  #3,D0
              DC.W   0xAB1D
```

```
                MOVE.L  (SP)+,D0
        }
}

/* DisposeGWorld -
 */
pascal void DisposeGWorld(offscreenGWorld)GWorldPtr offscreenGWorld;{
        asm {
                MOVE.L offscreenGWorld(A6),-(SP)
                MOVEQ  #4,D0
                DC.W   0xAB1D
        }
}

/* GetGWorld -
 */
pascal void GetGWorld(port, gdh)CGrafPtr *port;GDHandle *gdh;{
        asm {
                MOVE.L port(A6),-(SP)
                MOVE.L gdh(A6),-(SP)
                MOVEQ  #5,D0
                DC.W   0xAB1D
        }
}

/* SetGWorld -
 */
pascal void SetGWorld(port, gdh)CGrafPtr port;GDHandle gdh;{
        asm {
                MOVE.L port(A6),-(SP)
                MOVE.L gdh(A6),-(SP)
                MOVEQ  #6,D0
                DC.W   0xAB1D
        }
}

/* CTabChanged -
 */
pascal void CTabChanged(ctab)CTabHandle    ctab;{
        asm {
                MOVE.L ctab(A6),-(SP)
                MOVEQ          #7,D0
                DC.W           0xAB1D
        }
}

/* PixPatChanged -
 */
pascal void PixPatChanged(ppat)PixPatHandle       ppat;{
        asm {
                MOVE.L ppat(A6),-(SP)
                MOVEQ  #8,D0
                DC.W   0xAB1D
        }
}

/* PortChanged -
 */
pascal void PortChanged(port)GrafPtr port;{
        asm {
                MOVE.L port(A6),-(SP)
                MOVEQ  #9,D0
                DC.W   0xAB1D
        }
}
```

```
/* GDeviceChanged -
 */
pascal void GDeviceChanged(gdh)GDHandle gdh;{
	asm {
		MOVE.L gdh(A6),-(SP)
		MOVEQ  #0x0A,D0
		DC.W   0xAB1D
	}
}

/* AllowPurgePixels -
 */
pascal void AllowPurgePixels(pm)PixMapHandle pm;{
	asm {
		MOVE.L pm(A6),-(SP)
		MOVEQ  #0x0B,D0
		DC.W   0xAB1D
	}
}

/* NoPurgePixels -
 */
pascal void NoPurgePixels(pm)PixMapHandle pm;{
	asm {
		MOVE.L pm(A6),-(SP)
		MOVEQ  #0x0C,D0
		DC.W   0xAB1D
	}
}

/* GetPixelsState -
 */
pascal GWorldFlags GetPixelsState(pm)PixMapHandle pm;{
	asm {
		SUBQ.L #4,SP
		MOVE.L pm(A6),-(SP)
		MOVEQ  #0x0D,D0
		DC.W   0xAB1D
		MOVE.L (SP)+,D0
	}
}

/* SetPixelsState -
 */
pascal void SetPixelsState(pm, state)PixMapHandle pm;GWorldFlags state;{
	asm {
		MOVE.L pm(A6),-(SP)
		MOVE.L state(A6),-(SP)
		MOVEQ  #0x0E,D0
		DC.W   0xAB1D
	}
}

/* GetPixBaseAddr -
 */
pascal Ptr GetPixBaseAddr(pm)PixMapHandle pm;{
	asm {
		SUBQ.L #4,SP
		MOVE.L pm(A6),-(SP)
		MOVEQ  #0x0F,D0
		DC.W   0xAB1D
		MOVE.L (SP)+,D0
	}
}

/* NewScreenBuffer -
```

```
 */
pascal QDErr NewScreenBuffer(globalRect, purgeable, gdh, offscreenPixMap)
       Rect                  *globalRect;
       Boolean                      purgeable;
       GDHandle              *gdh;
       PixMapHandle   *offscreenPixMap;{

       asm {
              SUBQ.L #2,SP
              MOVE.L globalRect(A6),-(SP)
              MOVE.B purgeable(A6),-(SP)
              MOVE.L gdh(A6),-(SP)
              MOVE.L offscreenPixMap(A6),-(SP)
              MOVEQ  #0x10,D0
              DC.W   0xAB1D
              MOVE.W (SP),D0
       }
}

/* DisposeScreenBuffer -
 */
pascal void DisposeScreenBuffer(offscreenPixMap)PixMapHandle offscreenPixMap;{
       asm {
              MOVE.L offscreenPixMap(A6),-(SP)
              MOVEQ  #0x11,D0
              DC.W   0xAB1D
       }
}

/* GetGWorldDevice -
 */
pascal GDHandle GetGWorldDevice(offscreenGWorld)GWorldPtr offscreenGWorld;{
       asm {
              SUBQ.L #4,SP
              MOVE.L offscreenGWorld(A6),-(SP)
              MOVEQ  #0x12,D0
              DC.W   0xAB1D
              MOVE.L (SP)+,D0
       }
}
```

•••

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: Re: How do I get the slot of the main screen?

In article <334@6sigma.UUCP> blm@6sigma.UUCP (Brian Matthews) writes:

> >I can't figure out how to get the slot number of the main screen.

Re-read the graphic device chapter of Inside Mac Vol 5. Them write code similar to:

```
    GDHandle curgd;

    if(world->hasColorQD){
        for(curgd = GetDeviceList();curgd != NIL;curgd = GetNextDevice(curgd)){
            if(TestDeviceAttribute(curgd, screenDevice) &&
                TestDeviceAttribute(curgd, screenActive) &&
                TestDeviceAttribute(curgd, mainScreen)){
                    return curgd;            /* found it */
                }
            }
        }
    }
}
```

--- David Phillip Oster

●●●

From: austing@Apple.COM (Glenn L. Austin)
Subject:Re: Drawing to an off screen bitmap (with code)

In article <8912070531.AA07560@cadman.nyu.edu> deragon@CADMAN.NYU.EDU (John Deragon) writes:

> Hi folks,
> I did a small animation program on my Macintosh Plus,
>it steps through drawing each frame. The problem is, it is damn slow.
>So, faced with the fact that I cant afford a Mac IIcx, I would like to
>know how I could draw frames to an offScreen Bitmap. I am using LSC 4.0.

Drawing into an off-screen bitmap is really quite easy, if you follow a few rules:

1) The rowBytes *MUST* be even, otherwise the odd rows will be on an odd byte, just begging for address error!
2) Create a GrafPort for the new bitmap by calling OpenPort on a newly created GrafPort structure.
3) Save the current port and restore it after drawing to the off-screen port.

Some sample code (off the top of my head) in C:

```
BitMap  bm;  /* my off-screen bitmap */
GrafPort gp; /* my "off-screen grafport */
GrafPtr  oldPort; /* the current grafport */

GetPort(&oldPort); /* get the current grafport */
bm.rowBytes = 20; /* 160 pixels / 8 = 20 bytes */
SetRect(&bm.bounds, 0, 0, 160, 160); /* set the bounds (remember l,t,r,b!) */
bm.baseAddr = NewPtrClear(20 * 160); /* create a new ptr and zero the mem */
if (bm.baseAddr != nil)
{
  OpenPort(&gp); /* create a new, temporary grafport */
  SetPortBits(&bm); /* set the portBits to our bitmap */
  BlockMove((Ptr) &bm.bounds, (Ptr) &gp.portRect, sizeof(Rect)); /* copy the
           rect from bounds to portRect */
  RectRgn(gp.visRgn, &bm.bounds); /* change the visRgn to match */
  SetPort(&gp); /* use our port */

  /* Draw in the port just like you would draw on the screen! */

  SetPort(oldPort); /* restore the old port before disposing our ours! */
  ClosePort(&gp); /* so that the port memory is released *
  /* The bitmap is still around and valid and contains a bitmap of your */
  /* drawing */

  DisposPtr(bm.baseAddr); /* dispose of our off-screen bitmap memory */
}
```

It couldn't be easier! In fact, if you specify the port returned from the PrOpenPage with a SetPort, you are actually drawing into an off-screen bitmap that gets sent to the printer!

Glenn L. Austin

●●●

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: Re: Color Quickdraw and copybits glitches (ps to colorqQD with Code)

Once again, here is a short example of using color quickdraw. This program takes a text file called "bush.ps" which is a postscript 16-gray level bitmap of president bush, reads it in and displays it in a window. This example is missing a few resources, and the actual "bush.ps" file.

```
/* ShowBush.c - short program to display a postscript picture of Bush.
     in color quickdraw. You also get Bush on your clipboard.
   by David Phillip Oster.

   Copy and use this in any way you see fit.

    Basically an example of the pain you have to go through to actually
    use color quickdraw.

    In old quickdraw, you had some data and a window, and you CopyBit()s the data to
        the window.

    In color quickdraw, you have as your source:
        a.) the data.
        b.) a color table.
        c.) a "pixmap" that connects the data to the color table.
    as your destination, you have:
        a.) a color window
        b.) a palette, that informs the window manager of which colors you'll be
            using, so the color manager can change the color table in the video
            board to make those color available.

    This program works best with at least 16 colors!

 */
#include <ColorToolbox.h>

#define HEIGHT 200
#define WIDTH  200

#define NIL 0L
#define NOT !

SysEnvRec *world;
Handle         myPixels;
WindowPtr      myWin;
PixMapHandle   myPixMap;
CTabHandle         myCTable;
PaletteHandle  myPm;

enum {
    READERROR = 1,
    NEEDSCOLORQD,  /* this demo needs color quickdraw */
    NOPIXMAP,
    NOPIXELS,
    NOCTABLE,
    NOPALETTE,
    PIXDATACHANGED
};

enum {
    UNDOI = 1,
    COPYI = UNDOI + 1
};

main(){
    EventRecord myEvent;

    Init();
    for(;;){
        if(GetNextEvent(everyEvent, &myEvent)){
            HandleEvent(&myEvent);
```

```
        }
    }
}


Init(){
    PaletteHandle oldPm;

    FlushEvents(everyEvent, NIL);
    InitGraf( (Ptr) &thePort);
    InitFonts();
    InitWindows();
    InitCursor();
    InitDialogs(NIL);
    InitMenus();
    TEInit();

    world = (SysEnvRec *) NewPtr(sizeof(SysEnvRec));
    SysEnvirons(1, world);
    if( NOT world->hasColorQD){
        Error(NEEDSCOLORQD, "");
    }

    myWin = GetNewCWindow(128, NIL, -1L);                          /* set up the window */
    SetPort(myWin);
    ClipRect(&thePort->portRect);

    myPixMap = NewPixMap();                                                  /* allocate
the pixmap */
    if(NIL == myPixMap){
        Error(NOPIXMAP, "");
    }
    myPixels = NewHandle( ((long) HEIGHT/2)*WIDTH);              /* allocate the pixel data
*/
    if(NIL == myPixels){
        Error(NOPIXELS, "");
    }                                                                    /* allocate the
color table */
    myCTable = (CTabHandle) NewHandle(sizeof(ColorTable) + 15L*sizeof(ColorSpec));
    if(NIL == myCTable){
        Error(NOCTABLE, "");
    }
    InitPixels(myPixels);                                          /* set up the pixel data */
    InitCTable(myCTable);                                          /* set up the color table
*/

    (**myPixMap).bounds.left = 0;                            /* set up the pixmap */
    (**myPixMap).bounds.top = 0;
    (**myPixMap).bounds.right = WIDTH;
    (**myPixMap).bounds.bottom = HEIGHT;
    (**myPixMap).rowBytes = 0x8000 | (WIDTH/2);
    if(NIL != (**myPixMap).pmTable)
        DisposHandle((**myPixMap).pmTable);
    (**myPixMap).pmTable = myCTable;
    (**myPixMap).pixelSize = 4;

    myPm = NewPalette(16, myCTable, pmTolerant, 0);        /* set up the palette */
    if(NIL == myPm){
        Error(NOPALETTE, "");
    }

    if(NIL != (oldPm = GetPalette(thePort))){
        DisposePalette(oldPm);
    }
    SetPalette(thePort, myPm, TRUE);
    ActivatePalette(thePort);
```

```
}

/* SwapMove - copy from src to dest, moving scan lines from top to bottom
    as you go (postscript images are upside down compared to quickdraw images)
 */
SwapMove(src, dest, cnt)Ptr src, dest;long cnt;{
    register short i, j;

    for(i = 0;i <200;i++){
        BlockMove(src + (i*(WIDTH/2)), dest  + ((199-i)*(WIDTH/2)), WIDTH/2);
    }
}

/* InitPixels- initialize our handle to the bush data
    SwapMove since Postscript is upside down from Mac.
 */
InitPixels(h)Handle h;{
    register Ptr p, pMax;
    register short i;
    Handle h2, GetBush();

    h2 = GetBush();
    SwapMove(*h2, *h, GetHandleSize(h));
}


/* InitCTable - initialize our color table to uniform gray
 */
InitCTable(ct)CTabHandle ct;{
    short i;

    (**ct).ctSeed = GetCTSeed();
    (**ct).ctFlags = 0;
    (**ct).ctSize  = 15;
    for(i = 0;i < 16;i++){
        (**ct).ctTable[i].value = i;
        (**ct).ctTable[i].rgb.red =
        (**ct).ctTable[i].rgb.green =
        (**ct).ctTable[i].rgb.blue  = (unsigned) i * 16*257;
    }
}

/* Error - print the error message
 */
Error(i, s1)short i;Str255 s1;{
    Str255 s;

    GetIndString(s, 128, i);
    ParamText(s, s1, "", "");
    Alert(129, NIL);
    ExitToShell();
}

/* HandleEvent - quit on mouse or key event
 */
HandleEvent(event)EventRecord *event;{
    switch(event->what){
    case mouseDown:
    case keyDown:  ExitToShell();
    case updateEvt:   GoUpdate(event);      break;
    }
}

/* GoUpdate - handle update event, paste it to clipboard
 */
GoUpdate(event)EventRecord *event;{
```

```
    SetPort(event->message);
    BeginUpdate(thePort);
    CopyMyBits();
    EndUpdate(thePort);
}

/* CopyMyBits - put bits to screen, put bits to clipboard
 */
CopyMyBits(){
    Rect r, r2;
    PicHandle ph;

    HLock(myPixels);
    HLock(myPixMap);
    (**myPixMap).baseAddr = *myPixels;
    r.left = 0;
    r.top = 0;
    r.right = WIDTH;
    r.bottom = HEIGHT;
    r2 = r;
    r2.top *= 2;
    r2.left *= 2;
    r2.right *= 2;
    r2.bottom *= 2;

    ph = OpenPicture(&r);
    CopyBits(*myPixMap, &thePort->portBits, &r, &r, srcCopy, NIL);
    ClosePicture();
    CopyBits(*myPixMap, &thePort->portBits, &r, &r2, srcCopy, NIL);

    HUnlock(myPixMap);
    HUnlock(myPixels);

    ZeroScrap();
    HLock(ph);
    PutScrap(GetHandleSize(ph), 'PICT', *ph);
    SystemEdit(COPYI-1);  /* so multifinder will take the scrap */
    KillPicture(ph);
}

/* GetBush
 */
Handle GetBush(){
    OSErr  val;
    short  ref;
    Handle h;
    long   len;
    Str255 s;

    ref = 0;
    h = NIL;
    val = FSOpen("\pbush.ps", 0, &ref);
    if(noErr == val)val = GetEOF(ref, &len);
    if(noErr == val)h = NewHandle(len - 4);
    if(noErr == val && NIL != h){
        HLock(h);
        len = GetHandleSize(h);
        if(noErr == val)val = FSRead(ref, &len, *h);
        HexToBin(h);
        HUnlock(h);
    }
    if(0 != ref){
        FSClose(ref);
    }
    if(noErr != val){
        NumToString(val, s);
```

```
        Error(READERROR, s);
        if(NIL != h){
            DisposHandle(h);
            h = NIL;
        }
    }
    return h;
}

/* SkipHeader - advance p to point to data
 */
unsigned char *SkipHeader(p)register unsigned char *p;{
    while( NOT (p[0] == '\r' && p[1] == '0'))
        p++;

    return &p[1];
}

/* HexToBin - h is a locked handle containing text. turn it into binary
    Sorry about this. It is a quick and dirty hex ascii to binary routine.
 */
HexToBin(h)Handle h;{
    register unsigned char *src, *srcMax, *dest;

    src  = dest = (unsigned char *) *h;
    src = SkipHeader(src);
    srcMax = src + GetHandleSize(h);
    while(src < srcMax){
        if(*src >= '0' && *src <= '9'){
            *dest = (*src++ - '0') << 4;

            if(*src >= '0' && *src <= '9'){
                *dest++ |= *src++ - '0';
            }else if(*src >= 'A' && *src <= 'F'){
                *dest++ |= *src++ - 'A' + 10;
            }else if(*src >= 'a' && *src <= 'f'){
                *dest++ |= *src++ - 'a' + 10;
            }
        }else if(*src >= 'A' && *src <= 'F'){
            *dest = (*src++ - 'A' + 10) << 4;

            if(*src >= '0' && *src <= '9'){
                *dest++ |= *src++ - '0';
            }else if(*src >= 'A' && *src <= 'F'){
                *dest++ |= *src++ - 'A' + 10;
            }else if(*src >= 'a' && *src <= 'f'){
                *dest++ |= *src++ - 'a' + 10;
            }
        }else if(*src >= 'a' && *src <= 'f'){
            *dest = (*src++ - 'a' + 10) << 4;

            if(*src >= '0' && *src <= '9'){
                *dest++ |= *src++ - '0';
            }else if(*src >= 'A' && *src <= 'F'){
                *dest++ |= *src++ - 'A' + 10;
            }else if(*src >= 'a' && *src <= 'f'){
                *dest++ |= *src++ - 'a' + 10;
            }
        }else{
            src++;
        }
    }
    SetHandleSize(h, dest - (unsigned char *) *h);
}
```

•••

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: Re: Finding the size of the screen under Multifinder/Color QD (gdevices)

In article <10139@saturn.ucsc.edu> sirkm@ssyx.ucsc.edu (Greg Anderson) writes:

> >TN #117 says to look at the QuickDraw variable screenbits.bounds to
> >find the size of the screen, but on my Mac IIx, this rectangle is
> >always (0,0,0,0) when MultiFinder is installed (color or b&w).

Tim Maroney is right, it should return the bounding rectangle of the CRT that has the menu bar. It always has for me.

> >First question: what _is_ a gDevice? Is there one & only one gDevice
> >per monitor (/other display device) hooked up to the Mac, or might there
> >be multiple devices referencing a single monitor? If the later, why?

There should only be one at a time. Some video boards change the number of pixels they display depending on what mode they are in: i.e, they might be 640x480 in 8-bit per pixel mode, by 1024x1204 withe one bit per pixel. In addition, since graphic devices hold the inverse color table data structures (used for color matching when you do a CopyBits of a pixmap with one color table to a pixmap with a different table) there may be graphic devices that aren't connected to screens.

> >The reason I wish to know the size of the screen is simply to center my
> >dialogs on the screen. Given this purpose, should I just look at one
> >gDevice (namely, the one returned by GetGDevice)? Once I have this handle,
> >how should I determine the screen bounds? Get the PixMap from
> >(*gDev)->gdPMap and then look at pixMap.bounds?

use screenBits.bounds.

Here _is_ a use for gDevices:

I redefine the zoom box, so that windows zoom to the full size of the CRT they are on, rather than Apple's default. There is nothing more annoying than working with a two-monitor system, clicking on the zoom box of a window, and have it leap to the other, smaller montior.

Here is a code fragment:

```
/* GetBiggestZoom - set r to the dimensions of the largest
    extern SysEnvRec world;

    biggdh = NIL;
    if(world.hasColorQD){
        for(gdh = GetDeviceList(); gdh != NIL; gdh = GetNextDevice(gdh)){
            if(TestDeviceAttribute(gdh, screenDevice) &&
                TestDeviceAttribute(gdh, scrnActive)){
                if(biggdh == NIL || (**gdh).gdRect.bottom - (**gdh).gdRect.top >
                            (**biggdh).gdRect.bottom - (**biggdh).gdRect.top){
                    biggdh = gdh;
                }
            }
        }
    }
    if(biggdh == NIL){
        *r = screenBits.bounds;
        r->top += GetMBarHeight();
    }else{
        *r = (**biggdh).gdRect;
        if(TestDeviceAttribute(biggdh, mainScrn)){
            r->top += GetMBarHeight();
        }
    }
}
```

--- David Phillip Oster

•••

From: trebor@biar.UUCP (Robert J Woodhead)
Subject: Re: Bitmap not showing up in scrapbook (cliprect)

legg@sirius.ucs.adelaide.edu.au (Christian Legg) writes:

> A friend of mine is having problems with displaying a bitmap that he has
>saved in the scrapbook. He uses copybits to copy the bitmap into a PICT
>resource which he then saves into the scrapbook. The scrapbook indicates
>in the lower right hand corner that the item is a PICT, but does not display
>it.

Thou hast been bitten by that most egregious of monsters, the Immense ClipRect. When you create the grafport that you draw into to generate the picture, always set it's cliprect to just surround the area that will be drawn. Otherwise you get a huge cliprect (-32768,-32768,32767,32767) and the scrapbook and other programs go a little bonkers because of it.

--
Robert J Woodhead, Biar Games, Inc. !uunet!biar!trebor I trebor@biar.UUCP

•••

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: **Finding the size of the screen under Multifinder/Color QD**

In article <10139@saturn.ucsc.edu> sirkm@ssyx.ucsc.edu (Greg Anderson) writes:

>>>TN #117 says to look at the QuickDraw variable screenbits.bounds to
>>>find the size of the screen, but on my Mac IIx, this rectangle is
>>>always (0,0,0,0) when MultiFinder is installed (color or b&w).

In article <9429@hoptoad.uucp> tim@hoptoad.UUCP (Tim Maroney) writes:
>>Uh, I really don't think so. Check again.

In article <1102@urbana.mcd.mot.com> willcox@urbana.mcd.mot.com
(David A Willcox) writes:
>This sounds a bit like what happened when I tried to use screenBits
>from within an XCMD. The problem, of course, was that I hadn't called
>InitGraf() (I hope that's the right name) from within my XCMD, and I
>was seeing a local copy of screenBits that had not been initialized.
>(I'm using Think C 3.0.)

When I first read this, I though, "Of course!" But now, I'm not sure this makes sense. The XCMD should be called using HyperCard's A5, and its QD globals are initialized already. Maybe the problem is that LSC has a bug that looks for the QD globals off A4 if you are linking a code resource? That wouldn't be too surprising. But the LSC code resource glue doesn't mess about with the caller's A5, which should remain constant throughout.

>I tried calling InitGraf() within my XCMD, and that gave me valid
>values for screenBits, but things rapidly went to hell on return to
>HyperCard, since HyperCard's quickdraw globals didn't work any more.
>(Obvious, in hindsight.)

Sort of. This seems to be compatible with the A4-bug hypothesis,assuming that the code generated was using offsets from A4 and you passed a pointer to thePort which was also A4-relative.

>Can anyone tell me, is there a better way to get at global variables
>such as screenBits from within an XCMD?

Tell me, did you try the ugly way of accessing QD globals from assembly language, from IM I-163? I bet it would work.
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

•••

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject: Re: Finding the size of the screen under Multifinder/Color QD (with C code)

In article <10164@saturn.ucsc.edu> sirkm@ssyx.ucsc.edu (Greg Anderson) writes:

> >I know it's illegal to call InitGraf() from within XCMD's, but can I
> >call it from within a DA?

No, you can't call it from a DA either.

> >If I cannot call InitGraf() from within my DA, then how can I find the
> >size of the screen?

Here is how you access the quickdraw globals from a non-application, assuming that _some_ application has set them up by calling InitGraf: (A safe assumption for D.A.s, CDEFs, MDEFs, WDEFs, CDEVs, & FKEYs)

```
/* QuickDraw - access to globals from a code resource
       brought to you by David Phillip Oster
       (oster@well.sf.ca.us  or oster@dewey.soe.berkeley.edu)
       You may use this as you see fit.
 */
typedef struct QuickDraw {
       LongInt       randSeed;
       BitMap screenBits;
       Cursor arrow;
       Pattern       dkGray;
       Pattern       ltgray;
       Pattern       gray;
       Pattern       black;
       Pattern       white;
       GrafPtr       thePort;
} QuickDraw;


/* DisableSelf - this is from a CDEF I wrote, it dims the control.
 */
DisableSelf(ch)ControlHandle ch;{
       QuickDraw *qd;
       Rect r;

       qd = (QuickDraw *) ( *(Byte **) CurrentA5 - (sizeof(QuickDraw) -
sizeof(GrafPtr)) );
       r = (**ch).contrlRect;
       PenMode(patBic);
       PenPat(qd->gray);     /* <-- Note, a QD global! */
       PaintRect(&r);
       PenNormal();
}
```

David Phillip Oster

•••

From: tim@hoptoad.uucp (Tim Maroney)
Subject: **Re: Marquee Help? (with Code)**

In article <18304@dartvax.Dartmouth.EDU> sean@eleazar.dartmouth.edu (Sean P. Nolan) writes:

> >I saw on here recently a passing mention that someone was using code that
> >produced the MacPaint-like Marquee around a selection. The Johnny Carson theme
> >comes to mind. But I have been trying, off-and-on, to figure out how to do
> >just that, given a selection rectangle and a nifty slot to do its thing in

>my Event loop.
>
>If anybody has the ah-ha insight on how to do this, I'd really appreciate a
>note. Currently I shoved the problem aside and just InvertRect the selection,
>which is uuuuuuuugly.

Might as well just post dwb's code that I got from a local BBS:

```
;
; (c) Copyright 1987, David W. Berry
;
               case       object
               string     pascal

               load    'MacStuff'

               export     MarqueeRect, MoveMarqueeRect

patterns       record     entry,decr
marqueePat        dc.l       $0f87c3e1, $f0783c1e
shiftPat       dc.l       $88442211, $88442211
               endr

;
;   Frame rectangle, using marquee pattern.  The rect is drawn in xor mode.
;   This procedure should be called to draw the marquee initially, then
;   MoveMarqueeRect should be called repeatedly to move the pattern.
;   Finally, call MarqueeRect again to erase the marquee.  The display
;   will end up the same as it was before.
;

MarqueeRect       proc       export

Frame          record     {a6link},decr
rect           ds.l       1
return         ds.l       1
a6link         ds.l       1
state          ds.b       psRec
locals         equ            *
               endr

               with       Frame

               link       a6,#locals
               pea            state(a6)
               _GetPenState

               _PenNormal

               pea            patterns.marqueePat
               _PenPat

               move.w     #patXor,-(a7)
               _PenMode

               move.l     rect(a6),-(a7)
               _FrameRect

               pea            state(a6)
               _SetPenState

               unlk       a6
               rts

               endproc
```

```
;
;   Frame rectangle, using pattern that moves the marquee.  Then shift
;   the pattern and the marquee pattern, to keep the two in sync.
;   This procedure must be called repeatedly to achieve the effect of
;   motion.
;

MoveMarqueeRect   proc   export

Frame          record {a6link},decr
rect           ds.l   1
return         ds.l   1
a6link         ds.l   1
state          ds.b   psRec
locals         equ       *
               endr

               with   Frame

               link   a6,#locals

               pea       state(a6)
               _GetPenState

               _PenNormal

               pea       patterns.shiftPat
               _PenPat

               move   #patXor,-(a7)
               _PenMode

               move.l rect(a6),-(a7)
               _FrameRect

               pea       state(a6)
               _SetPenState

    ; shift the marquee pattern 1 slot to the left
               lea       patterns.marqueePat,a0
               bsr.s  rotate

    ; likewise the shift pattern
               lea       patterns.shiftPat,a0
               bsr.s  rotate

               unlk   a6
               rts

rotate:                move.l         (a0),d1
               move.l     4(a0),d0
               rol.l      #8,d0
               rol.l      #8,d1
               move.b     d0,d2
               move.b     d1,d0
               move.b     d2,d1
               move.l     d1,(a0)
               move.l     d0,4(a0)
               rts

               endproc

               end
```

End of quoted code marquee.a.

Here's MPW C code I use for the same effect. I'm afraid it's rather environment-dependent, but you should still be able to break it to your will.

```
static void RotatePatterns(Pattern marquee, Pattern shift)
{   unsigned char c;
    c = marquee[0];
    BlockMove(marquee + 1, marquee, 7);
    marquee[7] = c;
    c = shift[0];
    BlockMove(shift + 1, shift, 7);
    shift[7] = c;
}

static void
click(WindowPtr window, DocStorage **storage, EventRecord *event,
      Boolean *changed, Boolean *selected)
{   Point start, now; Rect r, r2; short tmp;
    DocInfo doc; Pattern marquee, shift; PenState pen;
#pragma unused (changed)
    unselect(window, storage);
    GetDocInfo(window, &doc);
    start = now = event->where;
    GetPenState(&pen);
    PenNormal();
    BlockMove((Ptr)(*storage)->marquee, (Ptr)marquee, 8);
    BlockMove((Ptr)(*storage)->shift, (Ptr)shift, 8);
    PenPat(marquee);
    PenMode(patXor);
    SetRect(&r, start.h, start.v, now.h, now.v);
    FrameRect(&r);
    while (WaitMouseUp()) {
        PenPat(shift);
        FrameRect(&r);
        RotatePatterns(marquee, shift);
        GetMouse(&now);
        SetRect(&r2, start.h, start.v, now.h, now.v);
        if (r2.left > r2.right)
            { tmp = r2.left; r2.left = r2.right; r2.right = tmp; }
        if (r2.top > r2.bottom)
            { tmp = r2.top; r2.top = r2.bottom; r2.bottom = tmp; }
        if (EqualRect(&r, &r2)) continue;
        PenPat(marquee);
        FrameRect(&r);
        FrameRect(&r2);
        r = r2;
    }
    if (*selected = !EmptyRect(&r))
        OffsetRect(&r, GetCtlValue(doc.hScroll), GetCtlValue(doc.vScroll));
    else { PenPat(marquee); FrameRect(&r); SetRect(&r, 0, 0, 0, 0); }
    BlockMove((Ptr)marquee, (Ptr)(*storage)->marquee, 8);
    BlockMove((Ptr)shift, (Ptr)(*storage)->shift, 8);
    (*storage)->selRect = r;
    SetPenState(&pen);
}

static void idle(WindowPtr window, DocStorage **storage)
{   Pattern marquee, shift; Rect r; PenState pen; DocInfo doc;
    r = (*storage)->selRect;
    if (EmptyRect(&r)) return;
    GetDocInfo(window, &doc);
    GetPenState(&pen);
    PenNormal();
    BlockMove((Ptr)(*storage)->marquee, (Ptr)marquee, 8);
    BlockMove((Ptr)(*storage)->shift, (Ptr)shift, 8);
    PenPat(shift);
```

```
    PenMode(patXor);
    OffsetRect(&r, -GetCtlValue(doc.hScroll), -GetCtlValue(doc.vScroll));
    FrameRect(&r);
    RotatePatterns(marquee, shift);
    BlockMove((Ptr)marquee, (Ptr)(*storage)->marquee, 8);
    BlockMove((Ptr)shift, (Ptr)(*storage)->shift, 8);
    SetPenState(&pen);
}
```

And in Rez, the patterns are:

```
resource 'PAT#' (33) { {
       $"0f 87 c3 e1 f0 78 3c 1e" ,
       $"88 44 22 11 88 44 22 11"
} };
```

--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: ari@eleazar.dartmouth.edu (Ari Halberstadt)
Subject: How to get the QD globals outside of an application

Several people have been curious about how to access the QuickDraw globals from within an XCMD or XFCN. Here's a little routine I wrote for THINK C which you can call when you initialize your XCMD or XFCN. If I remember correctly, MPW uses a single global structure to store the QuickDraw globals. Thus, one BlockMove would suffice to implement this routine in MPW.

```
/* QuickDraw global offsets off of A5 */
#define THEPORT      0             /* GrafPtr */
#define WHITE        THEPORT-8     /* Pattern */
#define BLACK        WHITE-8              /* Pattern */
#define GRAY         BLACK-8              /* Pattern */
#define LTGRAY           GRAY-8           /* Pattern */
#define DKGRAY           LTGRAY-8  /* Pattern */
#define ARROW        DKGRAY-68     /* Cursor */
#define SCREENBITS       ARROW-14  /* BitMap */
#define RANDSEED     SCREENBITS-4 /* long */

/* Setup the QuickDraw globals by getting them from their offsets off of A5 */
/* By Ari Halberstadt, 1990 */
static void
SetupQDGlobals()
{
    Ptr    p;

    p = *((Ptr *) CurrentA5);
    thePort = (GrafPtr) (THEPORT + p);
    BlockMove(WHITE + p, white, sizeof(Pattern));
    BlockMove(BLACK + p, black, sizeof(Pattern));
    BlockMove(GRAY + p, gray, sizeof(Pattern));
    BlockMove(LTGRAY + p, ltGray, sizeof(Pattern));
    BlockMove(DKGRAY + p, dkGray, sizeof(Pattern));
    BlockMove(ARROW + p, &arrow, sizeof(Cursor));
    BlockMove(SCREENBITS + p, &screenBits, sizeof(BitMap));
    randSeed = (long) (RANDSEED + p); /*I hope this line is correct...*/
}
```

--
Ari Halberstadt

●●●

From: lsr@Apple.COM (Larry Rosenstein)
Subject: Re: Marquee Help?

```
>PenMode(patXor);          /* I'm not sure this is either the correct
```

If you are doing this in a bitmap, then you should use patCopy. patXOR might look better in a structured graphics program, where there is always a lot of white space. (If you use patCopy then to erase the marquee you have to redraw the contents of the rectangle.0

```
>PenPat(patterns[patNum++]);
>patNum &= 3;
>FrameRect(pRect);                  /* Draw a rectangle */
>while (TickCount() - t < 4)        /* Wait a bit */
```

Another way to do this is to choose the pattern based on the value of TickCount % 4 (or TickCount % 8 if you use 8 patterns). The advantage is that the ants will move at the same speed regardless of the size of the rectangle. The disadvantage is that you may get jerky motion if your loop can't keep up.

--
Larry Rosenstein, Object Specialist

●●●

From: jmunkki@kampi.hut.fi (Juri Munkki)
Subject: Re: Finding the size of the screen under Multifinder/Color QD

In article <9530@hoptoad.uucp> tim@hoptoad.UUCP (Tim Maroney) writes:

```
>Nope.  The OpenPort strategy doesn't work on color systems, at least
>not in the simple way you seem to be suggesting.  If you just look at
>port.portBits.bounds, you'll be looking at something weird (a handle
>followed by two integers) on color systems.
```

Nope. You are confusing OpenCPort and OpenPort. OpenPort opens a regular grafport with a normal bitmap. I don't think that you bothered to look, but I did and even if I set my Mac II to 8 color, I get a rowBytes of 128 and a bounds of (0,0,480,640).

Creating a pixmap for offscreen grafports would have created an enormous amount of incompatibility. OpenPort fakes a bitmap that looks just as if you had a 1-bit screen. That's why the old MacPaint and SuperPaint painted all over the top of the screen. They got a rowBytes of 128 when it actually is 8 times that much. (Although the screen is 640 pixels wide, Apple throws away a few bytes and uses 1024 pixels per row of which only 640 are displayed.)

Juri Munkki jmunkki@hut.fi jmunkki@fingate.bitnet

●●●

From: beard@ux1.lbl.gov (Patrick C Beard)
Subject: Re: Finding the size of the screen under Multifinder/Color QD

In article <9530@hoptoad.uucp> tim@hoptoad.UUCP (Tim Maroney) writes:

```
#Nope.  The OpenPort strategy doesn't work on color systems, at least
#not in the simple way you seem to be suggesting.  If you just look at
#port.portBits.bounds, you'll be looking at something weird (a handle
#followed by two integers) on color systems.
#
#       if ((port->portBits.rowBytes & 0x8000) == 0)
#               screen = port->portBits.bounds;
#       else {  PixMapHandle pm = ((CGrafPtr) port)->portPixMap;
#               screen = (*pm)->bounds;
#       }
```

I've used the OpenPort strategy before and it does work. In fact, if you just open a classic GrafPort, you don't need to go through the tests you show above. I have also just used the portRect field of the GrafPort to give me the same information:

```
ScreenSize(Rect *r)
```

```
{
    GrafPort aPort;
    OpenPort(&aPort);
    *r = aPort.portRect;
    ClosePort(&aPort);
}
```

Another method is to just use the Window Manager port. Instead of the OpenPort call above, and instead of allocating a GrafPort on the stack you only need a GrafPtr.

```
ScreenSize(Rect *r)
{
    GrafPtr aPort;

    GetWMgrPort(&aPort);
    *r = aPort->portRect;
}
```

This seems to work on all systems I've tried.

Patrick Beard, Macintosh Programmer (beard@lbl.gov)

●●●

From: ech@cbnewsk.ATT.COM (Ned Horvath)
Subject: Re: Finding the size of the screen under Multifinder/Color QD

From article <10164@saturn.ucsc.edu>, by sirkm@ssyx.ucsc.edu (Greg Anderson):

> [quoted references to invalid screenbits.bounds deleted]
>
> I know it's illegal to call InitGraf() from within XCMD's, but can I
> call it from within a DA? It seems like I would have the same problems
> you experienced.
>
> If I cannot call InitGraf() from within my DA, then how can I find the
> size of the screen?
>
> >>Can anyone tell me, is there a better way to get at global variables
> >>such as screenBits from within an XCMD?
>
> Sort of -- make callbacks to HyperCard. This won't always get you what you
> want, but it can often be very helpful...

Assuming that the A5 you "inherit" is valid, and that InitGraf has been called by your host application (which had pretty well ALWAYS be true!), there are a couple of answers.

1. GrafPort myPort; OpenPort (&myPort);...your code here...ClosePort(&myPort); This gives you access to some of the variables you crave.

2. More generally, the low memory global CurrentA5 (long integer at 0x904) contains the address of the address of thePort (the last quickdraw variable) for the current application. So the following code snippet will get you a way to PEEK (never, never, NEVER poke!) at the QD globals:

```
struct qdvars {          /* note that the order is the reverse */
    long   randSeed;  /* of that on IM I-204 */
    BitMap screenBits;
    Cursor arror;
    Pattern    dkGray;
    Pattern    ltGray;
    Pattern    gray;
    Pattern    black;
    Pattern    white;
    GrafPtr    thePort;
} *myGlobals;

    myGlobals = (struct qdvars *) (
```

```
        (**(long**)0x904) -                 /* address of thePort */
        (long)&((struct qdvars *)0)->thePort  /* offset of thePort */
    );
```

I presume that if you're not using C you can figure out how to translate that nasty stuff into your own argot. The ugly part with 0x904 gets the address of thePort; the even more bizarre second term evaluates to a constant (at compile time) which is the offset of thePort in the full set of Quickdraw globals. You can now reference myGlobals->screenBits.bounds,
for example.

The second example works whenever the host application has called InitGraf(), even at interrupt time; most APPLs call InitGraf within micro seconds after launch, so you should be safe.

=Ned Horvath=

•••

From: oster@dewey.soe.berkeley.edu (David Phillip Oster)
Subject:Re: PICT resource to BitMap (with code)

In article <oZkwPji00WB5E_IkpG@andrew.cmu.edu> es2q+@andrew.cmu.edu (Erik Warren Selberg) writes:
>I'm having loads of problems trying to figure out how to grab a PICT
>resource, turn it into a bitmap, and then use CopyBits with it (actually, my
>problem is that I can't seem to turn said resource into a bitmap). Does

Write it yourself, it'll improve your mac programming skills, and there isno single system call to do it anyway. You'll do basically

```
BitMap *PictToBitMap(myPic)PicHandle myPic;{
    GrafPtr       savePort, myPort;
    GrafPort   myPortRec;
    static BitMap  myBitMap; /* so we can return it */
    Rect       r;

    GetPort(&savePort);
    myPort = &myPortRec;
    OpenPort(myPort);
    SetPort(myPort);
    myBitMap.bounds.top = myBitMap.bounds.left = 0;
    myBitMap.bounds.right = BIGENOUGHFORTHIS;
    myBitMap.bounds.bottom = ALSOBIGENOUGH;
    /* must be big enough an even. */
    myBitMap.rowBytes = ((myBitMap.bounds.right + 15) / 16) * 2);
    myBitMap.baseAddr = NewPtr(myBitMap.rowBytes * (long) myBitMap.bounds.bottom);
    SetPortBits(&myBitMap);
    /* next 2 lines necessary is PICT bigger than Mac's screen. */
    ClipRect(&myPort->portRect);
    CopyRgn(myPort->clipRgn, myPort->visRgn);

    r = (**myPic).picFrame;  /* 'cause it might move during draw */
    DrawPicture(myPic, &r);
    SetPort(savePort);
    ClosePort(myPort);
    return &myBitMap;
}
```

The above is from memory, and may have errors in it. The caller is responsible for deallocating my BitMap.baseAddr.

Handling color PICTs is left as an exercise to the reader. Oh, the above will _work_, but if you want the color table of the returned pixmap to be minimal (i.e., only as deep as necessary, with only the colors that are actually used by the PICT.) you either need to 1.) allocate a 32-bit deep grafport, draw the picture into that, then search its pixmap to see what colors it used, if it used less than 256 colors, but more than 16, build an 8 bit deep grafport, with an appropriate color table, and CopyBits from the 32-bit deep one to the 8-bit one, discard the 32-bit one, and return the 8. or 2.)

Write a parser for PICTs, search it for color tables and 16-bit or 32-bit deep pixmaps, and figure out the final color table yourself. This has the advantage that, at the cost of re-implementing color quickdraw yourself, it can be made to work even on a MacPlus, and the disadvantage that it is likely to break the next time Apple changes quickdraw. I would go with method 1, or just blow off the whole question and handle all color pixmaps as 32-bit deep.

David Phillip Oster

●●●

From: rcfische@polyslo.CalPoly.EDU (Raymond C. Fischer)
Subject: Re: Writing large PICT files???

In article <22163.25d040f6@kuhub.cc.ukans.edu> brownrigg@kuhub.cc.ukans.edu writes:
>Can someone describe for me how to create a PICT file consisting of only
>a bitMap?

This is from memory, so check the calls against Inside Mac.
Assuming the existence of a GrafPort that has your BitMap in it ...

```
SetPort (your GrafPtr);
OpenPicture
CopyBits (thePort^.portBits, thePort^.portBits,
      thePort^.portRect, thePort^.portRect,
      srcCopy, NIL);
ClosePicture
```

This will create a PICT that has nothing but the bitmap in it. To turn this into a PICT file, create a file of type PICT, write 512 bytes of zeros, then write entire handle contents returned from the OpenPicture/ClosePicture.

Make sure that at some time after opening the port and BEFORE calling OpenPicture, you set the ClipRgn of the port. One way to do this is

```
ClipRect (thePort^.portRect);
```

If you have only a BitMap and no GrafPort, then open a GrafPort, set the portBits to the bitmap record (using SetPortBits?), set the portRect (as in: thePort^.portRect := thePort^.portBits.bounds), and set the ClipRgn. The do the above CopyBits and then close the port;

If the GrafPort is a CGrafPort, this will produce a type 2 PICT (color) which won't work with any Mac that doesn't have color quickdraw.

Of course, GrafPort is synonymous with WindowRecord in this example.
Any questions?

Ray Fischer rcfische@polyslo.calpoly.edu

●●●

From: ephraim@think.com (Ephraim Vishniac)
Subject: Re: How's the screen cleaned after _SysError?

In article <6959@internal.Apple.COM> lsr@Apple.COM (Larry Rosenstein) writes:
>In article <34252@news.Think.COM> ephraim@Think.COM (Ephraim Vishniac)
>writes:
>> How is the screen restored after a call to _SysError?

>There's a low memory global called DSAlertRect. In GetMouse, if the high
>bit of DSWndUpdate is 1, then the system refreshes the rectangle stored in
>DSAlertRect.

GetMouse doesn't do this on my Mac II (6.02, Multifinder off). I disassembled it and found that the only subroutine call was to a journaling routine which exits immedidately if journaling is off.

Also, DSWndUpdate seems to be $FF most of the time (i.e., the high bit is always set).

>Since this doesn't seem to be documented, use at your own risk.

Not documented? Not so! Please refer to "Low Memory in Alphabetical Order" dated 12 Apr 85. (You did keep every scrap of paper from the Macintosh Software Supplements, right?) DSWndUpdate is at least as well documented as the famous MrMacHook. Notice the line that reads:

DSWndUpdate .EQU $15D ;01 GNE not to paintBehind DS AlertRect? [byte]

This suggests that it's GetNextEvent, not GetMouse, and painting will occur when the bit is cleared, not set. I haven't actually tried this out yet - it took me a while to find the right documentation.

--
Ephraim Vishniac ephraim@think.com ThinkingCorp@applelink.apple.com

●●●

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Optimizing Copybits (good Information with code)

[This is a repost of some comments of mine that appeared here last month.]

In Inside Mac V and in various places in the techNotes, it states that CopyBits is optimized for cases in which rowBytes is a multiple of four and in which the bitmaps are aligned. I recently did some testing on a (wide) sample of arbitrary bitmaps, and came up with the following interesting timings:

- Worst case ranges from about 102-105% of average case.
  (Which is to say: the average case is about as bad as you can get)
- Best case (perfect alignment) ranges from about 54-72% of average case.
  (i. e. Alignment has a *substantial* payoff.)

Remember, alignment (keeping the bit-index the same between corresponding bits in the source bitmap and the destination) is only useful under several conditions:

- You're always copying part of the source into one and only one corresponding part of the destination.
- The source and destination are at the same bitdepth, if you're working in color.

- For any given copy, the srcRect and destRect are of the same dimension (though not necessarily the same coordinates).

A good example of when alignment is useful: storing an offscreen bitmapcontaining the image to be painted into a window. Alignment will only change (and the offscreen map be recomputed) if the window is moved.

A good example of when alignment won't help at all: storing the image of a spaceship offscreen which can be copyBitted anywhere into the window. ("Anywhere"=any of 32 bit positions=alignment will only help you in 1/32 of your copies. Of course, you could store 32 differently-aligned spaceship copies offscreen, and choose the appropriate one to blit based on the dest rect.)

The following function might be useful for cases when you want to create an offscreen bitmap for copying to a window. Specify the window (which must already be created and in its proper location) in destWindow, and the bounds of the desired offscreen window (in the local coordinate system of destWindow). InitAlignedBitmap will generate the bitmap, tacking enough "slack" onto the left edge of the bitmap (i. e. by extending the bounds) to guarantee that any bit in theBits is aligned with the corresponding bit in the window's local coordinate system. This will guarantee you best case performance.

Lastly, a note or two:

- If User moves the window, you'll need to recompute and redraw the bitmap. You can dispose the old one and call this function again, if you like, or you can modify this one to SetPtrSize on the existing baseAddr

ptr. You may want to change the code to generate handles, instead of ptrs, but remember to lock and derefence them before drawing. Growing a window doesn't change the alignment (e. g. local [0,0] is still in the same place as it was pre-grow; therefore so is offscreen [0,0].)

- Color doesn't complicate things (multiply by pixDepth before determining rowBytes, and don't forget to set the high bit), but multiple screens and multiple pixel depths do. In that you can only be in alignment with one of the multiple screens at a time (or only *guarantee* that you'll be so), you'll have to choose between aligning for the screen which is deepest (which you'll want to do if your offscreen bitmap should be complexly colored), or for the screen which has the maximum destWindow area on it (which will optimize for speed over color fidelity). Once,you've figured out which GDevice to align to, plug in its pixMap for screenBits, below.

Enjoy.

```
function InitAlignedBitMap(var theBits:Bitmap;destWindow:WindowPtr):boolean;

{By Nick Jackiw}
{Allocates storage for a bitmap the bounds of which have been set before}
{calling.  The bitmap will be long-word aligned with the coordinate system}
{of the bit image which contains the destWindow.}

  var
   oldPort: GrafPtr;
   aPoint: point;
   bitOffset: integer;  {# of bits we need to expand the offscreen map by}

 begin
  GetPort(oldPort);     {Switch to desired port, for local->global conversion}
  SetPort(destWindow);
  aPoint := theBits.bounds.topLeft;
  LocalToGlobal(aPoint);

  bitOffset := (aPoint.h - ScreenBits.bounds.left) mod 32;
  if bitOffset < 0 then
   bitOffset := 32 + bitOffset;

  with theBits.bounds do
   left := left - bitOffset;{Expand dest bitmap by appropriate # of bits}

  with theBits do
   with bounds do
    begin
     rowBytes := ((right - left + 31) div 32) * 4;
     baseAddr := NewPtr(rowBytes * (bottom - top));
     InitAlignedBitMap := baseAddr <> nil
    end;
  SetPort(oldPort)
 end;
```

--
Nick Jackiw | Visual Geometry Project | Math Department

●●●

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: Fast Copybits suggestions needed, other questions

chris@ADMS-RAD.Unisys.COM (Chris Sterritt) writes:
> Someone a few weeks ago posted quite a long and informative post
> on making CopyBits go really fast. I didn't keep it then (in my almost
> infinite stupidity :-), so I'd like a copy now.

Other e-mail requests suggest to me that a repost would be welcome.

I'll post it separately.

> In case it doesn't answer these questions, I'll ask them now:
> (Background: I'm working on a game with animation, that I'd like
> to go as fast as possible).
>
> The game is (for now) black and white, will CopyBits do the
> right thing if it's run on a 4 or more bits deep screen (i.e., copying a
> 1-bit bitmap)? Or should I special-case the code for color?

Special-case. For every B&W pixel you draw on a four-bit screen, Color Quick- draw is going to have to convert it into a 4-bit pixel, and map it to the appropriate color. Though these might seem trivial, multiplied by a zillion pixels they amount to an incredible delay.

The cheesy thing to do would be to insist that user run in B&W mode. Better would be, at start-up, to allocate an offscreen pixmap at the current depth of the screen, draw all of your b&w bitmaps onto it, and then dispose the 1-bit bitmaps. Then use the curdepth pixmap as your library in the actual game. This keeps the task of translating depths isolated in your initialization code, not duplicated every time you draw an object onscreen.

--
Nick Jackiw

●●●

From:-Kelesi
Subject:How does one set up an offscreen buffer

>How does one set up an offscreen buffer so that CopyBits may be used to
>refresh the window when responding to an update event?

This is really a rather simple procedure. The idea is to draw into an offscreen port and then CopyBits the wanted information to the desired window. To setup an offscreen port for drawing, you can do something like this...

In Pascal (LSP):

```
Function SetupPort : GrafPtr;
var
  tmpPort, myPort : GrafPtr;
begin
  GetPort(tmpPort);
    { Need to allocate memory for the port }
  myPort := GrafPtr(NewPtr(sizeof(GrafPort));
    { -- Error checking for nil pointer goes here -- }
    { Create the internal structures for the port }
  OpenPort(myPort);
    { Now we need to set aside enough memory to draw into }
  with myPort^.portBits do
    begin
      baseAddr := NewPtr(rowBytes * (bounds.bottom - bounds.top));
        { -- Error checking for nil pointer goes here -- }
        { Don't need to initialize any of the rest of the fields in }
        { the port because Quickdraw has done all we need already. }
    end;
  { And return the resulting port, voila, an offscreen graph port! }
  SetPort(tmpPort);
  SetupPort := myPort;
end;
```

or, for them (LS)C buffs...

```
#define myBM myPort->portBits
GrafPtr SetupPort()
{
    GrafPtr tmpPort;
    register GrafPtr myPort;
```

```
    GetPort(&tmpPort);
    myPort = (GrafPtr) NewPtr(sizeof(GrafPort));
        /* -- Error checking for nil pointer goes here -- */
        /* Create the internal structures for the port */
    OpenPort(myPort);
        /* Now we need to set aside enough memory to draw into */
    myBM.baseAddr = NewPtr(myBM.rowBytes * (myBM.bounds.bottom -
        myBM.bounds.top));
        /* -- Error checking for nil pointer goes here -- */
        /* And return the resulting port, voila, an offscreen graph port! */
    SetPort(tmpPort);
    return myPort;
}
#undef myBM
```

There are several problems with the program as it stands above. 1) it does not have any error conditions. 2) It eats up about 6K of the heap everytime it's called (or more if your doing color, but if that's the case, you'll _need_ to look at Macintosh Technote #120)

Notice that I saved the current port on entry, this is because I didn't really want SetupPort to change my current graph port. If you don't care, you can remove these lines (but you may never know when you may need them, I spent several hours tracking down a single missing SetPort once...not a pleasant experience.)

You can save memory by passing a window to the procedure and then resizing the port to the window size. If you do this though, you have to resize the port everytime you resize the window, something of a bother. Enough of this, let's describe how to draw, etc.

The procedure to draw into the newly created port may look something like this (in Pascal):

```
Procedure DrawInMyPort( port : GrafPtr );
var
  tmpPort : GrafPtr;
begin
  GetPort(tmpPort);
  SetPort(port);
    { various Quickdraw calls here }
  SetPort(tmpPort);
end;
```

And in C...

```
DrawInMyPort(port)
GrafPtr port;
{
    GrafPtr tmpPort;

    GetPort(&tmpPort);
    SetPort(port);
        /* various Quickdraw calls here */
    SetPort(tmpPort);
}
```

Pretty easy huh? The way the port is setup, you have the entire 'screen' to draw into, we'd have to do a few more calls if the window are 'larger' than the screen (e.g. paint programs). Finally, we want to do an update event, drawing from the offscreen port to a window:

```
Procedure UpdateWind( wind : WindowPtr; port : GrafPtr );
var
  tmpPort;
begin
  GetPort(tmpPort);
  SetPort(wind);
  BeginUpdate(wind);
  CopyBits(port^.portBits, wind^.portBits, port^.portRect,
    wind^.portRect, srcCopy, nil);
```

```
  EndUpdate(wind);
  SetPort(tmpPort);
end;
```

or

```
#define NIL (0L)
UpdateWind(wind, port);
WindowPtr wind;
GrafPtr port;
{
    GrafPtr tmpPort;

    GetPort(&tmpPort);
    SetPort(wind);
    BeginUpdate(wind);
    CopyBits(&port->portBits, &wind->portBits, &port->portRect,
        &wind->portRect, srcCopy, NIL);
    EndUpdate(wind);
    SetPort(tmpPort);
}
```

These are all pretty much basic routines here but they should work. If you have any more questions, you might refer to the Macintosh technical notes. (You might be able to find the most recently distributed tn's on comp.mac.sources (comp.sources.mac?)) Specifically, TN#120 (which I know was uploaded recently) Hope this helps!

-Kelesi

•••

From:Written by John Olsen to go CCW, optimized by Mike Morton, and unknown
Subject: Rotate Bit Map CCW (full source code)

```
int RotateBMapCCW(srcBMap, dstBMap)
    BitMap *srcBMap, *dstBMap;
/* Given a source and destination bitMap, rotate it CCW 90°.
 * Bitmap can be of any size, will also align on word boundaries.
 * Written by John Olsen, optimized by Mike Morton, adapted to C by me.
 *
 * This routine has a little problem. The width of the destBMap is based
 * on its rowBytes, not its bounds rect, for speed reasons… but this
 * pretty much guarantees garbage in the no-man's land between the
 * bounds.right and the rowBytes. This causes a problem when counting
 * pixels in the caller routine, so the caller must take pains to
 * reclear this area.
 *
 * Returns FALSE if memory allocation problems, else returns TRUE.
 *
 * See MacTutor, V.4-N.11-p.86 for more details.
 */
{   int        colmCount;
    long    currentWord;
    long    lowerLeft;
    int        wordCount;

    asm
    {   movem.l    d3-d7/a2-a5,-(sp)        ; save registers

        ; Initialize destination bitMap, etc.
        ;    a0 = result of _NewPtr call            | d0 = size of bitMap allocate
        ;  a1 = ptr to srcBMap                  | d1 = width / 2 (source)
        ;  a2 = ptr to dstBMap                  | d2 = height / 2 (source)
        ;  a3 = dest bits                       | d3 = left | top
        ;  a4 = copy of dest bits                  | d4 = ctrPt.h
        ;  a5 = not used                        | d5 = ctrPt.v
```

```
        move.l srcBMap(a6),a1                   ; point to the source BitMap
        move.l dstBMap(a6),a2                   ; point to the destination BitMap

        ; Compute height of source BitMap rounded up to nearest word.
        move.w bds + bo(a1),d7                  ; bottom into d7
        sub.w      bds + to(a1),d7              ; bottom - top = height, put in d7
        move.w d7,d0                            ; save copy of height for later

        ; rnd = (height + 15) / 16, follows:
        addi.w #15,d7                           ; add #15 to height, put in d7
        lsr            #4,d7                       ; divide by 16
        lsl            #1,d7                       ; Multiply by 2 = rowBytes of dstBMap

        ; d7 now contains rowBytes of dstBMap. Now compute the dstRect for
        ;  dstBMap.
        ;
        ;  Compute the center point of srcRect, note the center points are
        ;  not necessarily the same in the vertical direction relative to
        ;  srcBMap becuz of rounding that follows, (i.e., height of srcBMap
        ;  is directly proportional to rowBytes of dstBMap).
        lsr            #1,d0                       ; divide height by 2
        move.w d0,d2                            ; make a copy of height/2 for later
        move.w bds + to(a1),d5                  ; get 'top' of srcRect
        add.w      d0,d5                        ; d5 contains vert compnt of ctr pt

        ; Compute the width of srcRect & put in d4
        move.w bds + ri(a1),d3                  ; get right and put in d3
        sub.w      bds + le(a1),d3              ; right - left = width, put in d3
        lsr            #1,d3                       ; divide width by 2
        move.w d3,d1                            ; make a copy of width/2 for later
        move.w bds + ri(a1),d4                  ; get right of srcRect
        sub.w      d3,d4                        ; d4 contains horiz compnt ctr pt

        ; Now get center point and compute bounds for dstBMap.
        ;  Height and width are reversed now for the 2 BitMaps.
        ;
        ;      left       := ctrPt.h - height / 2;
        ;      top            := ctrPt.v - width / 2;
        ;      right      := ctrPt.h + height / 2;
        ;      bottom := ctrPt.v + width / 2;
        ;
        ;  Put data in direct to avoid overhead of SetRect.
        ;  Compute 'top'.
        move.w d5,d3                            ; get a copy of ctrPt.v and put in d3
        sub.w      d1,d3                        ; ctrPt.v - width / 2 = 'top' in d3
        swap       d3                           ; move 'top' to hiword

        ; Compute 'left'.
        move.w d4,d3                            ; get a copy of ctrPt.h & put in loword
d3
        sub.w      d2,d3                        ; ctrPt.h - height / 2 = 'left' in loword
d3

        ; We now have 'top' in hiword & 'left' in loword d3.
        ;  Compute 'bottom'.
        move.w d5,d6                            ; get a copy of ctrPt.v in d3
        add.w      d1,d6                        ; ctrPt.v + width / 2 = 'bottom' in d3
        swap       d6                           ; move 'bottom' to hiword

        ; Compute 'right'.
        move.w d4,d6                            ; get a copy of ctrPt.h and put in loword
d6
        add.w      d2,d6
        ; We now have 'bottom' in hiword & 'right' in loword d6.
        ;
        ; Assign data to dstBMap structure directly.
```

```
        move.w d7,rB(a2)                    ; put in rowBytes of dstBMap
        move.l d3,bds + topLOff(a2)         ; put the coord pair in direct
        move.l d6,bds + botROff(a2)         ; put the coord pair in direct
        move.w d7,d0                        ; put rowBytes in d0
        mulu       rB(a1),d0                ; multiply rowBytes src*dst = d0
        lsl           #3,d0                    ; multiply d0 by 8 for size of dstBMap
        _NewPtr CLEAR                       ; allocate size of dstBMap (a2 pts to it)
        cmp.l      #0,a0                    ; succesful?
        bne           @cont                    ; yes, continue
        clr.l      d0                       ; no, return FALSE & …
        return;                                ; …exit
cont:
        move.l a0,bA(a2)                    ; move a0 to baseAddr field of dstBMap

        ;---------------------------------------------------------------------------
        ; a0 = not used                        | d0 = rowBytes of srcBMap
        ; a1 = points to bits of srcBMap       | d1 = width / 2 (source)
        ; a2 = ptr to dstBMap                     | d2 = height / 2 (source)
        ; a3 = dest Bits                       | d3 = current word scrBMap
        ; a4 =                                    | d4 = copy of width of srcBMap
        ; a5 = not used                        | d5 = current dstBMap
        ; a6 = not used                        | d6 = insideLoop count (current bit)
        ; a7 = not used                        | d7 = rowBytes of dstBMap
        ;---------------------------------------------------------------------------
        ; ****************************************************************
        ; *
        ; *           Start of actual rotation of bitMaps
        ; *
        ; ****************************************************************
        ; Assumptions:
        ; height of destination bitMap = rowBytes of srcBMap
        ;

        ; initialize d0 to contain the pointer to the srcBMap
        move.w rB(a1),d0                    ; get rowBytes of srcBMap

        ; get regs pointing to bits
        move.l (a1),a1                         ; a1 pts to BITS of srcBMap
        move.l (a2),a3                         ; a3 pts to BITS of dstBMap

        ; Compute the lower left corner of the dstBMap - this maps to
        ; the upper left corner of the srcBMap and will be used as base
        ; from which to offset into the dstBMap when we need to set a bit.
        clr.l      d1                       ; clear d1
        move.w d0,d1                        ; load rowBytes of srcBMap
                                            ; (rowBytes of srcBMap = ht of dstBMap)
        lsl.w      #3,d1                    ; multiply by 8 to get ht of dstBmap when
rot'd
        mulu.w d7,d1                        ; multiply by rowBytes(dstBMap) = #words in
        sub.w      d7,d1                    ; dstBMap less one row
        add.l      a3,d1                    ; add the computed offset to the baseAddr of
dstBMap
        move.l d1,lowerLeft                 ; init lower left

        ;---------------------------------------------------------------------------
        ;   a0 = not used                    |   d0 = rowbytes of srcMap
        ;   a1 = points to bits of src bitMap |   d1 = SCRATCH
        ;   a2 = currentWord                 |   d2 = colmCounter
        ;   a3 = dest bits                      |   d3 = current srcMap word
        ;   a4 = SCRATCH                     |   d4 = colmBitLoop counter
        ;   a5 = not used                    |   d5 = current word destMap
        ;   a6 = not used                    |   d6 = insideLoop count (current bit)
        ;   a7 = not used                    |   d7 = rowBytes of dest bitMap
        ;---------------------------------------------------------------------------
        ;
```

```
        ; init wordCount to zero
        ;
        move.w #0,wordCount                 ; set wordCount to one before we start
        ;
        ; init colmLoop counter
        ;
        move.w d7,d2                        ; get rowBytes(dest)
        lsr            #1,d2                   ; divide by 2 - d2 now contains the outer
counter
        move.w d2,colmCount                 ; put the max value away for reference
        moveq.l    #0,d2                       ; set d2 to zero since we use it for the
counter

colmLoop:
        ;
        ; this is the outside loop
        ;
        move.w #15,d4                           ; init colmBitLoop counter to 15
        move.l lowerLeft,a2                     ; a2 is currentWord, init it

colmBitLoop:
        ;
        ; this is the loop that does the bit counting in the destMap - it is really a
        ; misuse of the 'dbra' instruction since it is not an independent loop but it
        ; is a quite cheap method of doing our counting
        ;

        cmp.w      #7,d4                        ; are we doing high byte or low?
        bgt            @doByte1                    ; skip if high byte
        addq       #1,a2                        ; else point to low byte

doByte1:
rowLoop:

        ;
        ; loop across each row in the srcMap and keep count each time we do a row
        ; this 'rowCount' will used to compute the offset from 'lowerLeft'
        ;
        move.w (a1)+,d3                     ; get a word to rotate into d3
        addq.w #2,wordCount                 ; add 1 for each word ---jdo---

        add.w      d3,d3                    ; test next bit in the current source word
        bcc.s      @noBitToChange              ; if bit's not set, skip to next
        bset       d4,(a2)                     ; set proper bit in high or low byte in
currentWord

noBitToChange:
        sub.l      d7,a2                    ; subtract rowBytes(destMap) from current word

        ; Do the same as above, fifteen more times.
        add.w      d3,d3
        bcc.s      @1
        bset       d4,(a2)
@1  sub.l      d7,a2

        add.w      d3,d3
        bcc.s      @2
        bset       d4,(a2)
@2  sub.l      d7,a2

        add.w      d3,d3
        bcc.s      @3
        bset       d4,(a2)
@3  sub.l      d7,a2

        add.w      d3,d3
```

```
        bcc.s       @4
        bset        d4,(a2)
@4  sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @5
        bset        d4,(a2)
@5  sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @6
        bset        d4,(a2)
@6  sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @7
        bset        d4,(a2)
@7  sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @8
        bset        d4,(a2)
@8  sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @9
        bset        d4,(a2)
@9  sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @10
        bset        d4,(a2)
@10     sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @11
        bset        d4,(a2)
@11     sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @12
        bset        d4,(a2)
@12     sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @13
        bset        d4,(a2)
@13     sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @14
        bset        d4,(a2)
@14     sub.l       d7,a2

        add.w       d3,d3
        bcc.s       @15
        bset        d4,(a2)
@15     sub.l       d7,a2

        ;
        ; at this point we've just completed a word from the srcMap
        ;
        cmp             wordCount,d0                ; cmp wordCount to rowBytes-are we done with
a row ?
        bne             @rowLoop                    ; no - do another word
```

```
        cmp.w       #7,d4                   ; did we do high byte or low?
        bgt             @dobyte2                ; skip if high byte
        subq        #1,a2                   ; else undo the fudge to a2
dobyte2:

        ;
        ; we just completed a row so add 2 to the count
        ;
        move.w #0,wordCount                 ; ---jdo--- reset wordCount to zero

        ;
        ; the following use of dbra keeps track of our destMap bit for us
        ;
        move.l  lowerLeft,a2                ; re-init currentWord
                                            ; set the currentWord to the lowerLeft(current)
                                            ; lowerLeft is moved over (right) by a word each
                                                                ; time we do a row in the
srcMap
        dbra        d4,@colmBitLoop         ; go thru the bits (15->0) in the
                                            ; destMap - then exit and do it
                                                ; again for rowBytes/2 times

        ;
        ; were done counting thru the bits of a destMap word - at this point we have
        ; actually finished a column in the destMap. The column is a multiple of
        ; 16 bits wide (and being at least 1 colm) and as tall as the destMap is
        ; (which is also equal to rowBytes of the srcMap)
        ;
        addq.l #2,lowerLeft             ; this moves us into the next
                                        ; 'column' of bits in the destMap
        addq.w #1,d2                    ; add 1 to the colmLoop counter
        cmp             colmCount,d2        ; are we done yet ?
        bne             @colmLoop           ; no - do it again
                                        ; else we're done

        move.l #1,d0                                ; return TRUE, all ok
        movem.l (sp)+,d3-d7/a2-a5           ; restore registers, bye!
    }
} /* RotateBMapCCW */


int RotateBMapCW(srcBMap, dstBMap)
    BitMap *srcBMap, *dstBMap;
/* Given a source and destination bitMap, rotate it CW 90°.
 * Bitmap can be of any size, will also align on word boundaries.
 * Written by John Olsen to go CCW, optimized by Mike Morton,
 * and adapted to C by me.
 *
 * Revised 2/26/89 by me to do CW rotation.
 *
 * This routine has a little problem. The width of the destBMap is based
 * on its rowBytes, not its bounds rect, for speed reasons… but this
 * pretty much guarantees garbage in the no-man's land between the
 * bounds.right and the rowBytes. This causes a problem when counting
 * pixels in the caller routine, so the caller must take pains to
 * reclear this area.
 *
 * Returns FALSE if memory allocation problems, else returns TRUE.
 *
 * See MacTutor, V.4-N.11-p.86 for more details.
 */
{   int         colmCount;
    long    currentWord;
    long    lowerLeft;
    int         wordCount;
    int         srcHeight;      /* pixel height of source bitMap */
```

```
asm
{   movem.l    d3-d7/a2-a5,-(sp)        ; save registers

    ; Initialize destination bitMap, etc.
    ;    a0 = result of _NewPtr call        | d0 = size of bitMap allocate
    ;  a1 = ptr to srcBMap                     | d1 = width / 2 (source)
    ;  a2 = ptr to dstBMap            | d2 = height / 2 (source)
    ;  a3 = dest bits                       | d3 = left | top
    ;  a4 = copy of dest bits                  | d4 = ctrPt.h
    ;  a5 = not used                              | d5 = ctrPt.v

    move.l srcBMap(a6),a1               ; point to the source BitMap
    move.l dstBMap(a6),a2               ; point to the destination BitMap

    ; Compute height of source BitMap rounded up to nearest word.
    move.w bds + bo(a1),d7                  ; bottom into d7
    sub.w      bds + to(a1),d7              ; bottom - top = height, put in d7
    move.w d7,d0                            ; save copy of height for later

    ; rnd = (height + 15) / 16, follows:
    addi.w #15,d7                           ; add #15 to height, put in d7
    lsr            #4,d7                       ; divide by 16
    lsl            #1,d7                       ; Multiply by 2 = rowBytes of dstBMap

    ; d7 now contains rowBytes of dstBMap. Now compute the dstRect for
    ;  dstBMap.
    ;
    ;  Compute the center point of srcRect, note the center points are
    ;  not necessarily the same in the vertical direction relative to
    ;  srcBMap becuz of rounding that follows, (i.e., height of srcBMap
    ;  is directly proportional to rowBytes of dstBMap).
    lsr            #1,d0                    ; divide height by 2
    move.w d0,d2                        ; make a copy of height/2 for later
    move.w bds + to(a1),d5              ; get 'top' of srcRect
    add.w      d0,d5                    ; d5 contains vert compnt of ctr pt

    ; Compute the height of srcRect & put in srcHeight - rcy 2/26/89
    move.w bds + bo(a1),d3              ; get bottom and put in d3
    sub.w      bds + to(a1),d3          ; bottom - top = height, put in d3
    move.w d3,srcHeight                     ; store here, need later…

    ; Compute the width of srcRect & put in d4
    move.w bds + ri(a1),d3                  ; get right and put in d3
    sub.w      bds + le(a1),d3              ; right - left = width, put in d3
    lsr            #1,d3                       ; divide width by 2
    move.w d3,d1                            ; make a copy of width/2 for later
    move.w bds + ri(a1),d4                  ; get right of srcRect
    sub.w      d3,d4                        ; d4 contains horiz compnt ctr pt

    ; Now get center point and compute bounds for dstBMap.
    ;  Height and width are reversed now for the 2 BitMaps.
    ;
    ;      left    := ctrPt.h - height / 2;
    ;      top         := ctrPt.v - width / 2;
    ;         right   := ctrPt.h + height / 2;
    ;      bottom  := ctrPt.v + width / 2;
    ;
    ;  Put data in direct to avoid overhead of SetRect.
    ;  Compute 'top'.
    move.w d5,d3                            ; get a copy of ctrPt.v and put in d3
    sub.w      d1,d3                        ; ctrPt.v - width / 2 = 'top' in d3
    swap       d3                           ; move 'top' to hiword

    ; Compute 'left'.
    move.w d4,d3                            ; get a copy of ctrPt.h & put in loword d3
```

```
        sub.w      d2,d3                        ; ctrPt.h - height / 2 = 'left' in loword d3

        ; We now have 'top' in hiword & 'left' in loword d3.
        ;  Compute 'bottom'.
        move.w d5,d6                               ; get a copy of ctrPt.v in d3
        add.w      d1,d6                        ; ctrPt.v + width / 2 = 'bottom' in d3
        swap       d6                           ; move 'bottom' to hiword

        ; Compute 'right'.
        move.w d4,d6                            ; get a copy of ctrPt.h and put in loword d6
        add.w      d2,d6                        ; ctrPt.h + height / 2 = 'right' in loword d6

        ; We now have 'bottom' in hiword & 'right' in loword d6.
        ;
        ; Assign data to dstBMap structure directly.
        move.w d7,rB(a2)                        ; put in rowBytes of dstBMap
        move.l d3,bds + topLOff(a2)      ; put the coord pair in direct
        move.l d6,bds + botROff(a2)      ; put the coord pair in direct
        move.w d7,d0                     ; put rowBytes in d0
        mulu       rB(a1),d0             ; multiply rowBytes src*dst = d0
        lsl            #3,d0                ; multiply d0 by 8 for size of dstBMap

        _NewPtr CLEAR                    ; allocate size of dstBMap (a2 pts to it)
        cmp.l      #0,a0                 ; succesful?
        bne            @cont                ; yes, continue
        clr.l      d0                    ; no, return FALSE & …
        return;                             ; …exit
cont:
        move.l a0,bA(a2)             ; move a0 to baseAddr field of dstBMap

        ;-------------------------------------------------------------------------------
        ; a0 = not used                                 | d0 = rowBytes of srcBMap
        ; a1 = points to bits of srcBMap                | d1 = width / 2 (source)
        ; a2 = ptr to dstBMap                              | d2 = height / 2 (source)
        ; a3 = dest Bits                                 | d3 = current word scrBMap
        ; a4 =                                             | d4 = copy of width of srcBMap
        ; a5 = not used                                 | d5 = current dstBMap
        ; a6 = not used                                 | d6 = insideLoop count (current bit)
        ; a7 = not used                                 | d7 = rowBytes of dstBMap
        ;-------------------------------------------------------------------------------
        ; ********************************************************************
        ; *
        ; *            Start of actual rotation of bitMap
        ; *
        ; ********************************************************************
        ; Assumptions:
        ; height of destination bitMap = rowBytes of srcBMap
        ;

        ; initialize d0 to contain the pointer to the srcBMap
        move.w rB(a1),d0                    ; get rowBytes of srcBMap

        ; get regs pointing to bits
        move.l (a1),a1                          ; a1 pts to BITS of srcBMap

        ; The original CCW rotate started with the upper leftmost word of the
        ; srcBMap and went contiguously down thru memory. To go CW instead I have
        ; to do the opposite, start at the end and go up.
        ;
        ; The a3 reg points to the word to get next and in this case I must set it
        ; to the end, so what does the 'end' mean here? It means use the actual
        ; pixel height of the srcBMap, as determined by the bounds, multiplied by
        ; the srcBMap.rowBytes, added to the start, which will make a3 point to 1
        ; word past the end. Since the 1st fetch does a pre-decrement, this points
        ; us at the last real word, huzzah-huzzah…
        ;
```

```
        ; 1 prob, remains, the orginal CCW routine assumed
        ; - rcy 2/26/89
        clr.l       d1
        move.w d0,d1                        ; load rowBytes of srcBMap
        mulu.w srcHeight,d1                 ; times height of srcBMap = total wrds + 1
        add.l       d1,a1                   ; a1 now pts 1 word beyond end of srcBMap

        move.l (a2),a3                          ; a3 pts to BITS of dstBMap

        ; Compute the lower left corner of the dstBMap - this maps to
        ; the lower right corner of the srcBMap and will be used as base
        ; from which to offset into the dstBMap when we need to set a bit.
        clr.l       d1                      ; clear d1
        move.w d0,d1                        ; load rowBytes of srcBMap
                                            ; (rowBytes of srcBMap = ht of dstBMap)
        lsl.w       #3,d1                   ; multiply by 8 to get ht of dstBmap when rot'd
        mulu.w d7,d1                        ; multiply by rowBytes(dstBMap) = #words in
        sub.w       d7,d1                   ; dstBMap less one row
        add.l       a3,d1                   ; add the computed offset to the baseAddr of
dstBMap
        move.l d1,lowerLeft                 ; init lower left

        ;-------------------------------------------------------------------------------
        ;   a0 = not used                                       |   d0 = rowbytes of srcMap
        ;   a1 = points to bits of src bitMap   |   d1 = SCRATCH
        ;   a2 = currentWord                                    |   d2 = colmCounter
        ;   a3 = dest bits                                      |   d3 = current srcMap word
        ;   a4 = SCRATCH                                        |   d4 = colmBitLoop counter
        ;   a5 = not used                                       |   d5 = current word destMap
        ;   a6 = not used                                       |   d6 = insideLoop count (current
bit)
        ;   a7 = not used                                       |   d7 = rowBytes of dest bitMap
        ;-------------------------------------------------------------------------------
        ;

        ; init wordCount to zero
        ;
        move.w #0,wordCount                 ; set wordCount to one before we start
        ;
        ; init colmLoop counter
        ;
        move.w d7,d2                        ; get rowBytes(dest)
        lsr             #1,d2                   ; divide by 2 - d2 now contains the outer
counter
        move.w d2,colmCount                 ; put the max value away for reference
        moveq.l     #0,d2                       ; set d2 to zero since we use it for the
counter

colmLoop:
        ;
        ; this is the outside loop
        ;
        move.w #15,d4                       ; init colmBitLoop counter to 15
        move.l lowerLeft,a2                 ; a2 is currentWord, init it

colmBitLoop:
        ;
        ; this is the loop that does the bit counting in the destMap - it is really a
        ; misuse of the 'dbra' instruction since it is not an independent loop but it
        ; is a quite cheap method of doing our counting
        ;

        cmp.w       #7,d4                   ; are we doing high byte or low?
        bgt             @doByte1                ; skip if high byte
        addq        #1,a2                   ; else point to low byte
```

```
doByte1:
rowLoop:

        ;
        ; loop across each row in the srcMap and keep count each time we do a row
        ; this 'rowCount' will used to compute the offset from 'lowerLeft'
        ;
        move.w -(a1),d3                 ; get a word to rotate into d3 - rcy 2/26/89
        addq.w #2,wordCount             ; add 1 for each word ---jdo---

        lsr.w      #1,d3                ; test next bit in the current source word - rcy
2/26/89
        bcc.s      @noBitToChange           ; if bit's not set, skip to next
        bset       d4,(a2)                  ; set proper bit in high or low byte in
currentWord

noBitToChange:
        sub.l      d7,a2                ; subtract rowBytes(destMap) from current word

        ; Do the same as above, fifteen more times.
        lsr.w      #1,d3
        bcc.s      @1
        bset       d4,(a2)
@1  sub.l      d7,a2

        lsr.w      #1,d3
        bcc.s      @2
        bset       d4,(a2)
@2  sub.l      d7,a2

        lsr.w      #1,d3
        bcc.s      @3
        bset       d4,(a2)
@3  sub.l      d7,a2

        lsr.w      #1,d3
        bcc.s      @4
        bset       d4,(a2)
@4  sub.l      d7,a2

        lsr.w      #1,d3
        bcc.s      @5
        bset       d4,(a2)
@5  sub.l      d7,a2

        lsr.w      #1,d3
        bcc.s      @6
        bset       d4,(a2)
@6  sub.l      d7,a2

        lsr.w      #1,d3
        bcc.s      @7
        bset       d4,(a2)
@7  sub.l      d7,a2

        lsr.w      #1,d3
        bcc.s      @8
        bset       d4,(a2)
@8  sub.l      d7,a2

        lsr.w      #1,d3
        bcc.s      @9
        bset       d4,(a2)
@9  sub.l      d7,a2

        lsr.w      #1,d3
```

```
        bcc.s       @10
        bset        d4,(a2)
@10     sub.l       d7,a2

        lsr.w       #1,d3
        bcc.s       @11
        bset        d4,(a2)
@11     sub.l       d7,a2

        lsr.w       #1,d3
        bcc.s       @12
        bset        d4,(a2)
@12     sub.l       d7,a2

        lsr.w       #1,d3
        bcc.s       @13
        bset        d4,(a2)
@13     sub.l       d7,a2

        lsr.w       #1,d3
        bcc.s       @14
        bset        d4,(a2)
@14     sub.l       d7,a2

        lsr.w       #1,d3
        bcc.s       @15
        bset        d4,(a2)
@15     sub.l       d7,a2

        ;
        ; at this point we've just completed a word from the srcMap
        ;
        cmp             wordCount,d0                ; cmp wordCount to rowBytes-are we done with
a row ?
        bne             @rowLoop                    ; no - do another word

        cmp.w       #7,d4                       ; did we do high byte or low?
        bgt             @dobyte2                    ; skip if high byte
        subq        #1,a2                       ; else undo the fudge to a2
dobyte2:

        ;
        ; we just completed a row so add 2 to the count
        ;
        move.w #0,wordCount                     ; ---jdo--- reset wordCount to zero

        ;
        ; the following use of dbra keeps track of our destMap bit for us
        ;
        move.l  lowerLeft,a2                    ; re-init currentWord
                                                ; set the currentWord to the lowerLeft(current)
                                                ; lowerLeft is moved over (right) by a word each
                                                ; time we do a row in the srcMap
        dbra        d4,@colmBitLoop             ; go thru the bits (15->0) in the
                                                ; destMap - then exit and do it
                                                ; again for rowBytes/2 times

        ;
        ; were done counting thru the bits of a destMap word - at this point we have
        ; actually finished a column in the destMap. The column is a multiple of
        ; 16 bits wide (and being at least 1 colm) and as tall as the destMap is
        ; (which is also equal to rowBytes of the srcMap)
        ;
        addq.l #2,lowerLeft                     ; this moves us into the next
                                                ; 'column' of bits in the destMap
        addq.w #1,d2                            ; add 1 to the colmLoop counter
```

```
        cmp             colmCount,d2              ; are we done yet ?
        bne             @colmLoop                 ; no - do it again
                                                ; else we're done

        move.l #1,d0                            ; return TRUE, all ok
        movem.l (sp)+,d3-d7/a2-a5                 ; restore registers, bye!
    }
} /* RotateBMapCW */
```

•••

From: jmunkki@kampi.hut.fi (Juri Munkki)
Subject: Re: Aligning bitmaps? (with code)

In article <3791@ncsuvx.ncsu.edu> jnh@ecemwl.UUCP (Joseph N. Hall) writes:

> >What is the most straightforward procedure for ensuring that a window's
> >portBits begin on a word boundary? What I would like to do is check after
> >the user creates or resizes a graphics window and shift the window horizontally
> >(if necessary) so that it will be aligned for later updates from an offscreen
> >bitmap.
> >
> >Also, how much speed difference is there between an aligned and unaligned
> >one-to-one CopyBits? I assume that performance is no better for an
> >odd-byte-aligned bitmap than it is for a totally unaligned bitmap, right?

The speed difference is visible. I assume that you have an offscreen bitmap that you regularly copy into a window. The best solution for alignment is to shift the bitmap instead of forcing the window. You will only waste a few bytes for the extra margin necessary.

Here are a few code fragments:

```
/*
>>  AdjustVidPort adjusts the offscreen bitmap so that
>>  CopyBits can always work in 32 bit alignment.
>>
>>  Call AdjustVidPort whenever your window has moved.
>>
>>  The method was originally suggested by Larry Rosenstein
>>  from Apple's Advanced Technology Group.
*/
void    AdjustVidPort()
{
    BitMap temporary;
    GrafPtr    saved;

    GetPort(&saved);
    SetPort(&VidPort);

/*  Save old port location:
*/
    temporary=VidPort.portBits;

/*  Align port with magic formula:
*/
    MovePortTo((HMARGIN-VTWind->portBits.bounds.left) & 31,0);

/*  Align image:
*/
    CopyBits(  &temporary,          &VidPort.portBits,
           &VidPort.portRect,       &VidPort.portRect,
           0,srcCopy);

    SetPort(saved);
}
```

```
/*  Open an offscreen grafport for VT100 screen
**  Make it 32 pixels wider than absolutely necessary.
*/
    RowBytes=  ((ts.hsize*ts.charwidth+47)/16)*2;
...
    VidPort.portBits.bounds.left=0;
    VidPort.portBits.bounds.right=ts.hsize*ts.charwidth+32;
    VidPort.portBits.bounds.top=0;
    VidPort.portBits.bounds.bottom=ts.vsize*ts.lineheight;

    VidPort.portRect=VidPort.portBits.bounds;
    VidPort.portRect.right-=32;
    RectRgn(VidPort.visRgn,&VidPort.portRect);
    RectRgn(VidPort.clipRgn,&VidPort.portRect);
```

The above code makes sure that all your drawing is long word aligned. My terminal program doubled speed in cases where the window was badly positioned. All this has no effect on color screens, because I'm using a B&W bitmap even when the screen is in color. You would have to make small changes to apply the same technique to color pixmaps.

Juri Munkki jmunkki@hut.fi jmunkki@fingate.bitnet

●●●

From:Larry Rosenstein
Subject: Re: Aligning bitmaps?

In article <15288@dartvax.Dartmouth.EDU> earleh@eleazar.dartmouth.edu (Earle R. Horton) writes:

> > In article <3791@ncsuvx.ncsu.edu> jnh@ecemwl.UUCP (Joseph N. Hall) writes:
> > >What is the most straightforward procedure for ensuring that a window's
> > >portBits begin on a word boundary? What I would like to do is check after
> > >the user creates or resizes a graphics window and shift the window horizontally
> > >(if necessary) so that it will be aligned for later updates from an offscreen
> > >bitmap.
> >
> > I don't know what you mean by straightforward, but ANDing the
> > horizontal coordinate with NOT-0x0F before calling MoveWindow() should
> > work fine. Resize and create should not be a problem, but moving the

As someone noticed, Hypercard grids the window position and shows this by gridding the gray outline.

Another approach is to perform the alignment offscreen by allocating an offscreen bitmap that is 16 bits wider than necessary. Then you use a similar calculation to Earl's and align the offscreen buffer before copying it on the screen. I used this technique in the old MacApp Paint program and it worked very well. (And it looked much better than doing unaligned CopyBits.) In this case, you don't have to worry about non-standard window behavior.

I'm positive that alignment is important for 1-bit deep bitmaps on a 68000-based machine. (It might be that on a 68020 machine it's not necessary.) As I recall, it required 1/2 second on a MacPlus to align a MacPaint image offscreen. I did this by calling CopyBits with the same bitmaps as source and destination.

Also, notice that what counts is the global coordinates of the destination. If you use my trick of aligning the offscreen bitmap, then you will need to compute the global coordinates of your window. To do this first call SetPort(yourWindow), assign (0, 0) to a Point, and use LocalToGlobal.

Finally, I'm not sure that a word boundary is optimal alignment for Color Quickdraw. I think you might want to use longword alignment. I haven't done any stuff with offscreen pixmaps, so I don't know for sure.

> > and the screen depth is greater than one. Byte alignment seems to be
> > all that's necessary. Where did you get your information that word
> > alignment is needed?

I don't know about color, but for 1-bit deep images, word alignment alows QuickDraw to fetch and store longwords at a time (except for the edges). If you are only byte aligned then it may have to fetch things a byte at a time. (On

a 68000 you can't fetch and store a word on an odd boundary; on a 68020 you can fetch and store on odd byte boundaries, but you pay a speed penalty.)

> I don't notice any difference between aligned or non-aligned
> BitMaps with a black and white screen. On the other hand, these are
> subjective results since I haven't done formal timing.

On a MacPlus it is noticeable, although I haven't done any formal measurements, either.

Larry Rosenstein, Apple Computer, Inc. Object Specialist

●●●

From:earleh@eleazar.dartmouth.edu (Earle R. Horton)
Subject: Re: Aligning bitmaps?

In article <3967@internal.Apple.COM> lsr@Apple.COM (Larry Rosenstein) writes:
>In article <15288@dartvax.Dartmouth.EDU> earleh@eleazar.dartmouth.edu
>(Earle R. Horton) writes:
...
>Finally, I'm not sure that a word boundary is optimal alignment for Color
>Quickdraw. I think you might want to use longword alignment. I haven't
>done any stuff with offscreen pixmaps, so I don't know for sure.
...
>> I don't notice any difference between aligned or non-aligned
>> BitMaps with a black and white screen. On the other hand, these are
>> subjective results since I haven't done formal timing.
>
>On a MacPlus it is noticeable, although I haven't done any formal
>measurements, either.

OK, I broke out the old Timex, and actually timed some big screen animation on the Mac II using 1, 2, and 4 bits screen depth with big offscreen PixMaps. My conclusion after about a half hour of testing was that the only alignment that makes any difference to Color QuickDraw CopyBits() is longword alignment. The performance increase when both PixMaps are longword-aligned is what I would call spectacular, too. It's about 2 to 1! There does appear to be little, if any, performance increase for byte or word alignment over random alignment, so I wasn't totally wrong before.

Sorry for any confusion my earlier posting might have caused.

Earle R. Horton

●●●

From: awd@dbase.UUCP (Alastair Dallas)
Subject: ThePort and TENew (problem with font)

The problem that had me tearing my hair out was incredibly simple as it turned out. I can keep it to myself, or I can share it at the risk of appearing inexperienced. Here it is--you won't care now, but maybe this will settle into a niche at the back of your brain and when you trip over this problem, you'll think of me.

I created an edit field with a call to TENew() and I set some text into it. Then, at update time, I call TEUpdate(). It didn't display anything. I made sure that the port was set correctly at update time. Still nothing. I did a FrameRect() before the TEUpdate()--I got the rect, but no text. I dumped the TERec--perfect. I fiddled with the clipRgn. Finally, in desperation, I single-stepped through TEUpdate().

Do you know why I wasn't seeing text? The port must be correct at TENew() time, not TEUpdate(). You can be set to anything and still update your TEs, but you have to make sure the port is right when you create the TE. In most apps, you switch ports when you open windows and your calls to TENew() come after that, so everything's fine. In my case, the port was wrong at init time, but I fixed it up later.

Hope it helps somebody somewhere sometime.

/alastair/

•••

From: nicholas jackiw
Subject: Re: FatBits,thinbits and MapPt()

JON@wehi.dn.mu.oz (Jon Eaves) writes:

> I draw on the squash in the inverse colour (aaarrrgggh), 2 pixels get
> toggled horizontally and on the 'FatBits' version 1.5 fatPixels gets
> toggled.
>
> Why does this happen? I am pretty sure I am using MapPt() correctly
> and it works to draw the fatbits initially.
> ----------

Why don't you try posting your code? Sounds like an arithmetic error someplace, not a MapPt problem.

One thing you might want to consider doing, which is tremendously easy, is to do all drawing to just one of your two pictures, and then use CopyBits to scale it to the other. CopyBits is a whole lot faster than doing a bunch of FillcRects on your FatBits image, and all of the scaling is automated by quickdraw.

The one disadvantage is you can't display the "grid" in your fatbits image, unless you superimpose it after CopyBitsing.

-Nick
--
nicholas jackiw \jackiw%campus.swarthmore.edu@swarthmr.bitnet

•••

From: ftanaka@Apple.COM (Forrest Tanaka)
Subject: Re: 32-bit QuickDraw scaling behavior
Keywords: 32-bit Color QuickDraw scaling DrawPicture PenSize printing

In article <41276@apple.Apple.COM> ftanaka@Apple.COM (Forrest Tanaka) writes:

> The old characteristic of not scaling pen sizes when a picture was scaled
>was a bug in Quickdraw that's been fixed by 32-Bit Quickdraw. Since it's a
>bug fix, we thought that everyone would like it and no one would want to go
>back to the old way. So, there's no way right now to disable the fix aside
>from not using 32-Bit Quickdraw. It turned out that a few people depended on
>this bug, so there might be a way to disable pen size scaling in the future.
>I kind of doubt it, but it's not up to me.

Judging by the mail I've gotten, it looks like more people than I'd realized thought that this bug was pretty cool. So, I tried out something that I suspected might work for some people to fix the fix, and it does work.

If y'all turn to page 197 of Inside Macintosh I, you'll see a discussion about Quickdraw bottlenecks. By making your own bottlenecks, you can customize almost everything that Quickdraw does. When you call LineTo, StdLine is called to do the drawing. By providing your own routine to replace StdLine, you can make almost anything you want happen when LineTo is called. The same applies to all the others you see on the next couple of pages. Your StdLine replacement can even call StdLine, and it can even do things after calling StdLine--a non-breakable form of tail-patching if you want to think of it that way.

When you call DrawPicture, these bottlenecks are called to actually draw the contents of the Picture. So if the picture has a line in it, StdLine or your StdLine replacement will be called to draw the line. That's the key to fixing the fix.

What you can do is to create your picture in the usual way. Then before you call DrawPicture, replace the StdLine, StdRect, StdRRect, StdOval, StdArc, StdPoly, and StdRgn bottlenecks with your own bottlenecks. Except the StdLine case, the typical bottleneck will look something like:

```
if the verb is 'frame' then
   call PenSize(1,1)
call the standard bottleneck
```

That's it! It doesn't make sense for StdLine to take a verb, so you can just call PenSize(1,1) and then call the standard bottleneck in that case. What this is saying is that if the programmer or the picture calls Frame<X>, then set the pen size to 1 before calling the routine that actually draws the <X>.

The downside of this for some of you is that if you export your picture to another program, the pen sizes will be scaled again because they won't know to put in their own bottlenecks to set the pen size to 1 before drawing the objects. But, there is sort of a way around this even if you know what scale you want the result to be. In more pseudo-code, the operation could look something like this:

```
original picture = OpenPicture(small rectangle)
Draw some framed things
ClosePicture

Replace bottlenecks with the 'PenSize(1,1)' bottlenecks
scaled picture = OpenPicture (big rectangle)
DrawPicture (original picture, big rectangle)
ClosePicture
Set the bottlenecks back to what they were
```

Now, 'scaled picture' will contain a scaled version of 'original picture', but all the lines and frames will still have a pen size of 1 because you replaced the bottlenecks. Of course, if you scale 'scaled picture' larger, then the pen sizes will again be scaled unless you again replace the bottlenecks.

I doubt that this is practical in every case, but it sounds like some of you can do something like this.

Another option that I thought of as I was typing this is to patch PenSize. After giving it about three seconds of thought, I don't see what would be wrong with this. If any of you see a problem with this or the bottleneck method, or if you have a better way, please please please post it.

--
Forrest Tanaka Macintosh Developer Technical Support Apple Computer, Inc.

•••

From: lsr@Apple.COM (Larry Rosenstein)
Subject: Re: Need help creating a Quickdraw picture

In article <7366@jarthur.Claremont.EDU> mwilkins@jarthur.Claremont.EDU (Mark Wilkins) writes:
> Every time I've done it copying from a bitmap to itself, it's given me an
> empty picture definition.

Perhaps QuickDraw only records operations that involve the current grafPort's portBits. If you copy from one offscreen bitmap to another then the operation wouldn't involve the portBits. You might try installing your offscreen bitmap into a grafPort and use that port for creating the picture.

Larry Rosenstein, Apple Computer, Inc.

•••

From: kaufman@Neon.Stanford.EDU (Marc T. Kaufman)
Subject: Re: Still need help creating a QuickDraw picture

In article <2223@dftsrv.gsfc.nasa.gov> seiler@amarna.gsfc.nasa.gov writes:
... (stuff deleted)
>SetPort(windPtr);
>picHndl = OpenPicture(&(windPtr->portRect));
>CopyBits(&(windPtr->portBits), &(windPtr->portBits),
> &(windPtr->portRect), &(windPtr->portRect), srcCopy, NULL);
>ClosePicture();

>no copybits opcode shows up, but the sequence: version op, version, header op,
>size, fBBox, <4 bytes reserved>, DefHilite, Clip, opEndPic occurs.

Let me try (after all, everyone else has :-)). I don't see why you are changing ports. Why not just do the OpenPicture on your screen grafPort? The picture traps will catch everything.

If you really want to use another port, you need to make sure that the portRect, clipRgn, and visRgn are correct. You do NOT need to worry about the portBits.bounds, and indeed, you do not need any bitMap/Pixmap actually allocated for a port that is only used for capturing a picture.

Marc Kaufman (kaufman@Neon.stanford.edu)

●●●

From: pierce@radius.com (Pierce T. Wetter III)
Subject: Re: Need help animating with 24 bit card

>I wrote a program about 2 years back which needed to
>change the color of circles on the screen at a very fast rate. I
>accomplished this by calling AnimatePallette. Of course this limited
>me to a maximum of 254 objects at any time due to our 8 bit card. I
>now need to write a similar application which does the same thing except
>with many more objects... in the range of about a thousand.

>I thought it might be possible to do the same thing except with a
>24 bit card.

Funny you should ask about this, I just did this for PrecisionColor (radius'sMonitor calibration gizmo, that I am responsible for writing the software to)

You can animate colors on a 24 bit board, but there are some caveats:

1: You can only do 256 arbitrary colors at at time.
2: Other colors besides the one you wish to animate will animate as well.
3: For non arbitrary colors, you can animate 256 shades of red, 256 of blue and 256 of green.

The necessary technique is to use the gamma table to do your animation for you. All the 24-bit cards on the mac have an 8-bit by 8-bit table in front of the dac. One way to animate a color is to pick an entry in all three of these tables. Lets pick (0xFE,0xFE,0xFE) as an example. If I draw a circle in that color and then change those particular entries in the gamma table to whatever color I want, the circle will change to that color. (This is how the not-yet-released Turbo Precision Color works.)

This still limits you to 254 colors (assuming you don't animate full white and full black). However, you can do more by animating your circles to just red, and just green and just blue. If you do that, then you can use all the colors (N, 0, 0) & (0,N,0) and (0,0,N). If you don't mind your circles overlapping when you animate (for instance, you could have an indicator LED colored (N,N,0). You can then animate red to be indicating one variable, and green to be the other variable. The LED will then take on any shade from red to green to yellow. This lets you have more animating circles, but you can still only have 3x254=762 INDEPENDENT circles.

Of course, this probably won't help you any, but I had fun talking about what I was doing.

Pierce

P.S.: Since I mentioned it, its only fair to say that the new software is just a minor upgrade to support the 8..24 and 8..24GC. While I was at it I made it 3x faster, but that's beside the point.
--
My postings are my opinions, and my opinions are my own not that of my employer.

●●●

From: russotto@eng.umd.edu (Matthew T. Russotto)
Subject: Re: Setting up CLUTs - Advise???????

In article <24664.266d7b29@kuhub.cc.ukans.edu> brownrigg@kuhub.cc.ukans.edu writes:

>Can anyone suggest the general mechanism one needs to use to display a
>pixMap with PRECISELY CHOSEN colors?? I've read and re-read the relevent
>chapters of IMV5 and am still confused and frustrated as to what to do.

Yep. In fact, I can suggest TWO ways. One is to put a 'pltt' resource id 0 in your application resource file. The resource should contain valid pltt data with only two colors: black, and white. Then just use SetEntries from the color manager chapter to put YOUR colors in. Of course, be careful to SaveEntries and RestoreEntries at suspend and resume time under multifinder, so other applications don't get their colors messed up, and you don't have yours messed up. You also have to be very careful in a multi-screen environment using this method.

The second, and far superior method (IMO) is to use NewPalette to create a palette with the 254 colors you need, plus black and white in the first two entries (yes, I know some quantizers don't always put out a black and white. Tough.). Black and white should be marked as pmCourteous, all the other colors should be pmTolerant with a tolerance of 0 (pmIntolerant?). Then SetPalette(yourwindow, yourpalette, TRUE); ActivatePalette(yourwindow); to make the window automatically keep your color environment;

(the reason black and white should be set to pmCourteous is that the Palette manager used to reserve an additional index for them, in addition to the standard positions at the beginning and end of the color table. I don't know if this is still the case or if it is correct behavior, but setting them to courteous won't hurt)
--
Matthew T. Russotto russotto@eng.umd.edu

●●●

From: russotto@eng.umd.edu (Matthew T. Russotto)
Subject: Re: ? Bad pixmap data written by OpenPicture & CopyBits

In article <1990Jun13.181220.1@dev8j.mdcbbs.com> astor@dev8j.mdcbbs.com writes:

>My application has an offscreen pixmap, and I'm trying to copy the
>pixmap onto the scrap as a simple PICT.
>
>I use OpenPicture, CopyBits, ClosePicture, ZeroScrap, and PutScrap.

Try OpenPicture, CopyBits,ClosePicture,ZeroScrap,HLock,PutScrap.

--
Matthew T. Russotto russotto@eng.umd.edu russotto@wam.umd.edu

●●●

From: ccc_ldo@waikato.ac.nz (Lawrence D'Oliveiro, Waikato University)
Subject: Re: Fading ...(dissolve effect)

I had a play once with the idea of doing a "dissolve" effect using absolutely standard QuickDraw calls. It turns out that the CopyMask operation (Inside Mac page IV-24) is almost custom-designed for this sort of thing.

What I did was create a number of off-screen bitmaps, all the same size as the window in which the pictures will be displayed. The number of bitmaps corresponds to the number of "stages" in which the dissolve will take place: more stages makes the dissolve smoother, but takes longer.

If you look at a particular pixel position in all of these bitmaps, you will find that the corresponding bit is 1 in exactly one of the bitmaps, and zero in all the others. Thus the way you use them is by making repeated calls to CopyMask, each time specifying the complete new picture (drawn into an offscreen buffer) as the SrcBits argument, the PortBits for the window in which the display will appear as the DstBits, and a different one of the "stage" bitmaps as the MaskBits argument. The 1 bits in MaskBits select which pixels get copied from SrcBits to DestBits; once you've gone through all the stage bitmaps, the window will be displaying the complete new picture.

Of course, the distribution of 1 bits within each stage bitmap has to be reasonably random. The way I ensured this was, for each pixel position, to generate a random integer in the range 1 to the number of stages, and use that number to select the bitmap which would get the 1 bit (all the rest would get 0 for that pixel).

This random-number generation turned out to take quite a long time; to speed things up, I changed it to operate on just a 32*32 portion of the bitmap, and then simply replicated that bit pattern "cell" by "tiling" across and down with

CopyBits until I'd covered the entire bitmap. Another way would be to store the pregenerated 32*32 bit patterns in a resource and just read them into memory when the program runs.

Though I looked for it, I couldn't see any visible effect of repeating the same 32*32 pattern of bits--the distribution of the newly-appearing pixels still looked completely random. I didn't experiment to see if I could get away with a smaller cell size.

Another nice thing is that this technique also works with Color QuickDraw, and should also look good on a 24-bit-per-pixel display. For maximum speed, you should make SrcBits a PixMap with the same depth and colour table as DstBits, but MaskBits is still a plain, ordinary bitmap.

Speed? On 1-bit-per-pixel displays, with 6 dissolve stages, the result was quite presentable, even on a Mac Plus (though not as fast as HyperCard, which manages to use 12 stages). 8 bits per pixel on a vintage Mac II was actually slower than 1 bit per pixel on a Plus--but at least it worked!

Lawrence D'Oliveiro

●●●

From: h+@nada.kth.se
Subject: Re: Drawing all over the desktop

larsen@ginger.Princeton.EDU writes:

> ---
> I know that Apple doesn't want applications to draw all over the desktop.
> Since I'm doing it anyway, though, I need advice in order to do it as cleanly
> as possible. At the moment I'm checking screen depth and writing into video
> memory, but I would prefer a quickdraw solution.
> ---

The stuff to do is drawing into the WMgrPort. This is as compatible as it gets, though that isn't very. You have to be VERY careful to restore everything as it was before you call {Get/Wait}NextEvent. This means ony XORing, among other things. (Well, if it's a game and you don't want it to be MF compatible nor System 7 compatible, you don't have to restore the drawings, just the state of the WMgrPort)

I have written an INIT that, among other things, needs to draw in the WMgrPort, called SetWindow INIT. The source and the init is available via FTP from rascal.ics.utexas.edu, and is in the directory mac/hacking (if memory serves me right, might be programming or somewhere completely else...)

Generally, it is a VERY BAD idea to draw outside of windows. Only newly-converted MS-DOS programmers try and do this without a good reason. (And I mean GOOD)

Happy hacking,

h+

●●●

From: hildreth@cg-atla.agfa.com (Lon Hildreth)
Subject: Re: PICT hacking question
Keywords: PICT

In article <1732@mountn.dec.com> minow@bolt.enet.dec.com (Martin Minow) writes:

> >The original image was scanned at 300 dpi. When I look at the output
> >image (even without processing), it is clearly at a coarser resolution
> >(probably 72 dpi).
> >
> >Could anyone tell me how I get the actual image at its intended resolution
> >(both into my program and out to a new PICT file)? Inside-Mac and the
> >TechNotes stack don't seem to have anything relevant.

You need to put yourself in the StdBits bottleneck during playback. The hRes and vRes fields of the srcBits passed in contain the horizontal and vertical resolution of the image in dots per inch. They're Fixed point numbers, so you can call Fix2Long on them to get usable numbers. First, make sure that srcBits is a PixMap by checking the high bit of rowBytes.

For creating a new PICT, make sure that the hRes and vRes fields of your source pixmap is set properly. Also, the dest rect of the CopyBits call should be at 72 dpi.

--
Lon Hildreth ...!{decvax,uunet,samsung}!cg-atla!hildreth

●●●

From: keith@Apple.COM (Keith Rollin)
Subject: Re: Wanted- a window that comes up in black

In article <23026@dartvax.Dartmouth.EDU> anarch@eleazar.dartmouth.edu (The Anarch) writes:

> Apologies if this is an elementary question; I'm still pretty new to Mac
>programming. Anyway, I'm trying to get a window to come up onscreen with a
>black content region. Now I know that I can easily do NewWindow and then
>PaintRect the window's body, but that causes a brief white flash, which I'd
>like to avoid but have been unable to get rid of. What I *have* tried:
>changing the bkColor and bkPat for the window in various ways before a
>ShowWindow; I haven't gotten any positive results.

Here's one for the oneliners file:

The window Mgr doesn't use your window's bkPat when erasing the content region.

No. That would be too simple, too easy. Instead, since the content region is drawn in the WMgrPort, the Window Manager uses the WMgrPort's bkPat. Since this is always white, your window's content region is always drawn in white when it is updated.

There are several not-so-good ways around this. As I recall, if you have a color window, the window's rgbBkColor IS respected, and you can have your window updated in any color you want (but not pattern). You can also try munging around with the PaintWhite low-memory global, which can be used to prevent the window manager from clearing your content region to any color at all, allowing you to do it. However, there are problems with that, as Inside Mac I-297 points out. Finally, you could write a WDEF wherein the entire window is a structure region, and where there is no content region. Since the WDEF is solely responsible for drawing the strucRgn, you will have total control over how it gets redrawn.

--
Keith Rollin --- Apple Computer, Inc. --- Developer Technical Support

●●●

From: jwwalker@usceast.UUCP (Jim Walker)
Subject: Re: Wanted- a window that comes up in black

I thought of another "not so good way" that Keith Rollin didn't mention: Before showing your window, temporarily change the bkPat of the Window Manager port to black. Be sure to change it back to white once the window is drawn! You may also need to set the window's bkPat to black to prevent a flash. And you would probably need a custom WDEF that can draw its frame properly when the bkPat is black. (I did enough experimentation with this to convince myself that it could be made to work.)
--

Jim Walker jwwalker@cs.scarolina.edu 76367.2271@compuserve.com

●●●

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: Dragging Bitmaps?

siegel@endor.harvard.edu (Rich Siegel) writes:

> Does anyone have any suggestions on how to drag small bitmaps around, in the
> same fashion as MacPaint? Currently I drag outlines around, but it would be
> much nicer to move the object as a whole.
>
> R.

Sure: you'll need to use some offscreen bitmaps though. The major design decisions that affect the process are: (1) do you want it to be really smooth? (2) must your bitmaps be of eccentric shape or can they be limited to rects?

(1) For rectangles

The Cheap Way:

Get two offscreen bitmaps the size of the one to drag, and fill them both with the bitimages. Call these SOURCE and BUFFER. Now watch the mouse. Every time it moves, calculate the new location (presumably you measure the offset into the bitmap at which the mouse went down, and then calculate the new location based on this offset and the vector of mouse movement, so the mouse "sticks" to the same part of the rectangle). Copy BUFFER to the old location, copy the rect of "underlaid" bits at the new location to BUFFER, and then copy SOURCE to the new location. Repeat until mouseup.

The problem with this is that you'll have flicker--you'll be overwriting background bits with source bits and source bits with background bits.

The more expensive way:

Get an offscreen copy of the area in which you wish to permit dragging, without the dragable image in it. Call this BACKBITS. Now get an offscreen copy of the draggable image, call this SOURCE. Now get a third bitmap, the size of BACKBITS, left blank; call this SCRATCH. For each new mouse move, copy the UnionRect of the oldLocation and the newLocation from BackBits to Scratch, and then throw in (to Scratch) a copy of the SOURCE at new location. Copy this unionRect to the screen.

(2) For irregularly-shaped bitmaps

You know about BitMap2Rgn, I presume. It's on the DTS cds in object code, or you can count on it being available if you'll only run on 32Bit QD machines. (There may be licensing hassles concerning the object code; read the docs or write your own.) With BitMap2Rgn, get a region that includes/excludes the appropriate parts of your SOURCE. On a blank bitmap the size of SOURCE, do a FillRgn(bitsRgn,black). This gives you a mask, which you should use in either of the above schemes: wherever you formerly did CopyBits(source,somePlace...) now do a CopyMask(source, somePlace,maskBits,...).

If you want source code, I can whip something up. There are slightly more efficient ways to do it than the above, but that's the basic scheme as I understand it.

Call this BACKBITS.

--
nicholas jackiw I jackiw%campus.swarthmore.edu@swarthmr.bitnet

●●●

From: dowdy@apple.com (Tom Dowdy)
Subject: Re: Wanted- a window that comes up in black

Here's YET ANOTHER way to do this. You can judge if it is less or more terrible than those already suggested. I call it about even, but the one effect that it relies on IS documented.

I place it behind the WDEF approach, and ahead of hitting fields in the WMgrPort in the cleaniness rankings.

a) Open a port (PORT, not window) in the location where the window will be.
b) Fill port with black
c) Dispose of port.
d) Slam low memory PaintWhite ($9DC) to be zero (this keeps the window manager from filling the window with white, but leaves what is behind it, ie black), save the old copy.
e) Open your window, which will get the black area, and will NOT be filled with white.

f) Restore PaintWhite.

Tom Dowdy Internet: dowdy@apple.COM

•••

From: lsr@Apple.COM (Larry Rosenstein)
Subject: Re: Dragging Bitmaps?

In article <3398@husc6.harvard.edu> siegel@endor.harvard.edu (Rich Siegel) writes:
>
>Does anyone have any suggestions on how to drag small bitmaps around, in the
>same fashion as MacPaint? Currently I drag outlines around, but it would be
>much nicer to move the object as a whole.

The basic approach is to cache the image behind the moving bitmap, and use that cache to erase the bitmap before it moves. So the basic code would be to use 1 CopyBits to erase the old bitmap, and 1 CopyBits to draw the bitmap in its new position. (This assumes that you have cached the entire view beforehand. This is always better than copying the bits from the screen.)

You will get better results if you do both steps with 1 CopyBits. To do this you need another offscreen bitmap the size of the union of the old and new rectangles. You initialize this bitmap from the larger cache, and draw the moving bitmap in its desired position. Then you use 1 CopyBits to copy everything onto the screen at once.

--
Larry Rosenstein, Object Specialist

•••

From: beard@ux1.lbl.gov (Patrick C Beard)
Subject: Re: 2-16-256-million color mode
Keywords: standards, color, monitors

In article <1037@swan.ulowell.edu> jkeegan@hawk.ulowell.edu (Jeff Keegan) writes:
#Is there ANY "acceptible" way to change the color mode other than telling
#someone to go change it?

There is a new call that is documented in the latest batch of Tech Notes called "SetDepth()". This lets you do precisely this from software.

On the other hand, the user interface guidelines for this aren't really spelled out. I think that an application should notify the user that it is about to change the depth of the monitor, and allow the user to quit if that is not what he wants to do.

- Patrick Beard, Macintosh Programmer (beard@lbl.gov) -

•••

From: minow@mountn.dec.com (Martin Minow)
Subject: Reading PICT's revisited.

Thanks for all the good replies to my question on reading a 300 dpi PICT into a (72 dpi) window. Here's the code I use to read and write PICTs into an offscreen GrafPort.

Martin Minow
minow@bolt.enet.dec.com

```
/*                    ReadPicture.c                    */
/*
 * Picture I/O
 *
 * For some reason, the PICT dimensions do not correctly
 * describe the picture: it loses its 300 dpi resolution
 * when read in. MAGIC scales the PICT by a factor of 4.
```

```
 * We should get the "real" scale factor by inserting
 * our function into the StdBits bottleneck.
 */
#include "CleanPICT.h"
#include <SysErr.h>
#define MAGIC(size)  ((((long) size) * (4 * 72)) / 72)

/*
 * See TechNote 27 (original format), and the Essential
 * MacTutor, vol 3, p 417.  (This is now unused.)
 */
typedef struct {
  OSType  fType;          /* 'DRWG' for MacPaint files  */
  short    hdrID;          /* 'MD' for MacPaint format    */
  short    version;        /* File version                */
  short    prRec[60];      /* 120 byte print record      */
  Fixed    xOrigin;        /* Drawing origin              */
  Fixed    yOrigin;        /* Drawing origin              */
  Fixed    xScale;          /* Screen resolution          */
  Fixed    yScale;          /* Screen resolution          */
  short    atrState[31];    /* Drawing attribute state    */
  short    lCnt;            /* Top-level objects          */
  short    lTot;            /* Total number of objects    */
  long    lSiz;            /* Total size of list          */
  Long    top;            /* Box enclosing all objects  */
  Long    left;
  Long    bottom;
  Long    right;
  short    filler1[141];    /* 282 bytes unused           */
} MacDrawHdrRec;

static int      PICTfile;  /* The file ioRefNum          */

/*
 * Use this to peek into the bitmap as it is read in.
 * The peeking is done by jumping into the Debugger.
 */
pascal void      myStdBits(
    BitMap *, Rect *, Rect *, int, RgnHandle);
pascal void      read_picture_data(Ptr, int);
pascal void      write_picture_data(Ptr, int);

/*
 * read_picture() reads the picture from the PICT file,
 * constructing the window at the proper size.
 * "window" is really a document structure:
 *    typedef struct {
 *  WindowRecord  window;
 *  GrafPtr       pictPort; -- offscreen grafport
 *  Rect       pictSize; -- true PICT size
 *    } DocumentRecord;
 * The DOC macro handles type casting and dereferencing.
 */
OSErr
read_picture(window, theFile)
WindowPtr      window;    /* Read into this document  */
int            theFile;   /* From this open PICT file */
{
    PicHandle        handle;
    QDProcs          procedures;
    OSErr            status;
    long            size;
    long            place;
    Rect            box;
    GrafPtr          oldPort;
    MacDrawHdrRec  header;
```

```
    PICTfile = theFile;
    handle = (PicHandle) NewHandle(sizeof (Picture));
    if (handle == NIL) {
      DebugStr("\pCan't get memory for picture");
      return (MemError());
    }
    /*
     * Read the MacDraw header record -- that was a
     * good idea, but it didn't work as the headers
     * are garbage in the PICT I'm using.
     */
    if (sizeof header != 512)
      DebugStr("\pMacDrawHdrRec wrong size!");
    read_picture_data((Ptr) &header, sizeof header);
    HLock(handle);
    read_picture_data((Ptr) *handle, sizeof (Picture));
    HUnlock(handle);
    box = (**handle).picFrame;
    DOC.pictSize = box;
    box.right = MAGIC(box.right);
    box.bottom = MAGIC(box.bottom);
    GetPort(&oldPort);
    /*
     * Create an offscreen GrafPort.  See TechNote 41.
     */
    DOC.pictPort = CreateOSGrafPort(box);
    if (DOC.pictPort == NIL) {
      DebugStr("\pNo memory for picture");
      SetPort(oldPort);
      return (MemError());
    }
    SetStdProcs(&procedures);
    DOC.pictPort->grafProcs = &procedures;
    procedures.getPicProc = (Ptr) read_picture_data;
/*  procedures.bitsProc = (Ptr) myStdBits;    -- unused  */
    DrawPicture(handle, &box);
    DOC.pictPort->grafProcs = NIL;
    DisposHandle((Handle) handle);
    SetPort(oldPort);
    /*
     * Check for errors by getting the file position and
     * checking that it is at the end of file.
     */
    if ((status = GetEOF(PICTfile, &size)) != noErr
     || (status = GetFPos(PICTfile, &place)) != noErr) {
       DebugStr("\pCan't get EOF or file position");
       return (status);
    }
    if (size != place) {
      DebugStr("\pDidn't read entire picture");
      return (dsSysErr);
    }
    /*
     * Ok so far.  Now, change the window size so the
     * picture fills the window -- but keep the proportions
     * as close to the original as possible.
     */
    SetRect(
      &box,
      2,
      GetMBarHeight() * 2,
      width(DOC.pictSize) + 2,
      GetMBarHeight() * 2 + height(DOC.pictSize)
    );
    if (box.bottom > (screenBits.bounds.bottom - 2)) {
```

```
      box.bottom = screenBits.bounds.bottom - 2;
      size = height(box);
      box.right = box.left
        + ((long) width(box) * size) / height(DOC.pictSize);
    }
    if (box.right > (screenBits.bounds.right - 2)) {
      box.right = screenBits.bounds.right - 2;
      size = width(box);
      box.bottom = box.top
        + ((long) height(box) * size) / width(DOC.pictSize);
    }
    SizeWindow(window, width(box), height(box), TRUE);
    InvalRect(&window->portRect);
    ShowWindow(window);
    return (MemError());
}

/*
 * This should implement a "vanilla" StdBits -- for some
 * reason, though, it "bombs" when it runs to completion:
 * probably because its called with a NULL argument at
 * eof.  It was only used to check that the created
 * "MAGIC" bitmap is the same size as the bitmap inside
 * of the PICT -- by running the function under the
 * debugger.
 */
pascal void
myStdBits(srcBits, srcRect, dstRect, mode, maskRgn)
BitMap    *srcBits;
Rect      *srcRect;
Rect      *dstRect;
short      mode;
RgnHandle  maskRgn;
{
    CopyBits(
      srcBits,
      &thePort->portBits,
      srcRect, dstRect,
      mode,
      maskRgn
    );
}

/*
 * Called indirectly to read a chunk of picture data.
 */
pascal void
read_picture_data(data_ptr, byte_count)
Ptr        data_ptr;
int        byte_count;
{
    OSErr        status;
    long        count;

    count = byte_count;
    status = FSRead(PICTfile, &count, data_ptr);
    if (status != noErr)
      DebugStr("\pReading picture");
}

/*
 * write_picture() writes the current picture to a
 * specified (open) file.  It should be redone to
 * add error handling.
 */
void
```

```
write_picture(window, theFile)
WindowPtr      window;
int            theFile;
{
    PicHandle    picHandle;
    QDProcs      procedures;
    int          i;
    long        temp;
    Picture      header;
    GrafPtr      tempPort;
    GrafPtr      oldPort;

    GetPort(&oldPort);
    PICTfile = theFile;
    /*
     * Write a dummy MacPaint header
     */
    temp = 0L;
    for (i = 0; i < 512; i += sizeof temp)
      write_picture_data((Ptr) &temp, (int) sizeof temp);
    header.picSize = 0;
    header.picFrame = DOC.pictSize;
    write_picture_data((Ptr) &header, (int) sizeof header);
    /*
     * Write the picture by creating a GrafPort with the
     * same dimensions as the original, then drawing the
     * cleaned up picture.  Can this be done without
     * creating a new GrafPort?
     */
    tempPort = CreateOSGrafPort(DOC.pictSize);
    if (tempPort == NIL) {
      DebugStr("\pNo space for temp port");
      return;
    }
    SetStdProcs(&procedures);
    tempPort->grafProcs = &procedures;
    procedures.putPicProc = (Ptr) write_picture_data;
    picHandle = OpenPicture(&tempPort->portRect);
    CopyOSGrafPort(DOC.pictPort, tempPort);
    ClosePicture();
    KillPicture(picHandle);
    DeleteOSGrafPort(tempPort);
    DOC.pictPort->grafProcs = NIL;
    SetPort(oldPort);
}

/*
 * Called indirectly to write a chunk of picture data.
 */
pascal void
write_picture_data(data_ptr, byte_count)
Ptr        data_ptr;
int        byte_count;
{
    OSErr        status;
    long        count;

    count = byte_count;
    status = FSWrite(PICTfile, &count, data_ptr);
    if (status != noErr)
      DebugStr("\pWriting picture");
}
```

●●●

From: minow@mountn.dec.com (Martin Minow)
Subject: How to read a Pict into a scrollable window

In article <1990Jun29.215003.26596@Neon.Stanford.EDU> jhsu@Neon.Stanford.EDU (Jeffrey H. Hsu) writes:

> >Can somebody tell me how to read a PICT document into a scrollable window? I
> >can deal with the PICT as a resource but not as a file.

I can't help with the scrollable part, but here's some code I use to read a PICT file into an offscreen bitmap. (The code is incomplete, but it should give you the basic structure.) There's a Tech Note that explains what's going on.

```
extern int    picture_file;        /* PICT file, already open   */

void       read_background_picture(void);
pascal void   read_picture_data(Ptr, int);


/*
 * Called indirectly to read a chunk of picture data.
 */
pascal void
read_picture_data(data_ptr, byte_count)
Ptr        data_ptr;
int        byte_count;
{
    OSErr      status;
    long       count;

    count = byte_count;
    status = FSRead(picture_file, &count, data_ptr);
    if (status != noErr)
        status_error("\pReading background picture", status);
}

void
read_background_picture()
{
    PicHandle  handle;
    QDProcs       procedures;
    OSErr      status;
    long       size;
    long       place;
    int        picture_width, picture_height;
    int        screen_width, screen_height;
    Rect       box;

    SetStdProcs(&procedures);
    ((GrafPtr) overhead_window)->grafProcs = &procedures;
    /*
     * Use our function to read chunks of picture data
     */
    procedures.getPicProc = (Ptr) read_picture_data;
    handle = (PicHandle) NewHandle(sizeof (Picture));
    /*
     * Rewind the file and skip over the MacDraw header block.
     */
    if ((status = SetFPos(picture_file, fsFromStart, 512L)) != noErr) {
        status_error("\pCan't rewind picture file", status);
        return;
    }
    HLock(handle);
    /*
     * Start by reading the header information.
     */
    read_picture_data((Ptr) *handle, sizeof (Picture));
    HUnlock(handle);
    /*
     * Center the picture in the window and read it in.
```

```
     */
    box = (*handle)->picFrame;
    OffsetRect(           /* Normalize topleft to 0,0     */
        &box,
        -box.left,
        -box.top
    );
    screen_width = thePort->portRect.right - thePort->portRect.left;
    screen_height = thePort->portRect.bottom - thePort->portRect.top;
    picture_width = box.right - box.left;
    picture_height = box.bottom - box.top;
    OffsetRect(           /* Now, center it nicely */
        &box,
        (screen_width - picture_width) / 2,
        (screen_height - picture_height) / 2
    );
    DrawPicture(handle, &box);
    ((GrafPtr) overhead_window)->grafProcs = NIL;
    DisposHandle((Handle) handle);
    /*
     * Check for errors by getting the file position and
     * checking that it is at the end of file.
     */
    if ((status = GetEOF(picture_file, &size)) != noErr
     || (status = GetFPos(picture_file, &place)) != noErr)
         status_error("\pCan't get EOF or file position", status);
    if (size != place)
        status_error("\pDidn't read entire picture, lost", size - place);
}
```

Martin Minow

•••

Chapter 12 Resources & the Resource Manager

Creating a Resource Fork 200
Interesting problem whe GetResource('dctb',id) [***LONG***] 200
Puzzling Resource Problem 201
Interesting problem with GetResource('dctb', id) 201
Interesting problem with GetResource('dctb', id) 202
Altering the Resouce fork 202
OpenResfile Question 203
How to Openresfile [Summary] 203
OpenResfile Question 205

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: Creating a Resource Fork

In article <6956@hubcap.clemson.edu> mikeoro@hubcap.clemson.edu (Michael K O'Rourke) writes:

> >No, i don't just want to know how to create a resource fork. Obviously,
> >i'd just use CreateResFile. What I am trying to do is open the resource
> >fork for an existing file, but a file which currently doesn't have a
> >resource fork. Therefore, if i use OpenResFile or OpenRFPerm, they return
> >-1 because there isn't a resource fork. If i call CreateResFile, it thinks
> >I want to overwrite the existing file and it returns an error.

What error? I use CreateResFile to create resource forks for existing files and it works fine. This is actually the recommended way of using CreateResFile, due to some weirdness involving the Poor Person's Search Path. See some tech note or other (OK, I'll check -- it's 101). Of course, you have to frame it in volume-setting stuff, like:

```
GetVol -- get old volume
SetVol -- set volume of existing file
CreateResFile -- with name of existing file
SetVol -- back to old volume
```

--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: mjohnson@Apple.COM (Mark B. Johnson)
Subject: Re: Interesting problem whe GetResource('dctb',id) [*LONG***]**

The following is an original posting and followups to it concerning a problem with GetResource. It is long, so press 'n' now if you aren't interested. Posted on behalf of Jim Reekes, MacDTS Ethics Officer...

---
dolf,

Unfortunately, you've run across a patch to _GetResource that is there to solve a Dialog Manager problem. The dctb and actb are color tables for windows. This color table resource handle cannot be purged from memory, but all other Dialog Manager resources are supposed to be purged. Normally such resource DITLs, DLOGs, etc. are copied by the Dialog Manager. The problem is that the Dialog Manager didn't copy the dctb or actb, and this lead to dialog window's color table being purged.

So, a patched to _GetResource was put into place. When a 'dctb' or 'actb' is fetched, the Resource Manager performs a _HandToHand and returns this copy. This is bad because the Resource Manager has returned a valid handle that is not a resource, and calling _GetResInfo will fail with resNotFound! The work around suggested below works, but I do not recommend it.

```
SetResLoad(FALSE);
hDCTB := GetResource('dctb', foobar); { <-- this is bad, due to patch}
SetResLoad(TRUE);
LoadResource(hDCTB);
```

Setting ResLoad to false will not load the resource data but returns a valid handle that is empty. (The handle's master pointer is NIL.) When the patch code is called to perform the _HandToHand, the routine fails. The Memory Manager will set D0 to nilHandleErr when _HandToHand is passed an empty handle.

If you followed _GetResource with _MemError, you will get the result nilHandleErr. (BTW The Resource Manager isn't too consistent returning errors, and if you called _ResError you would get noErr.) Therefore, the above code does do what you wanted. It even looks as if there's nothing wrong, but it is not a good idea. It will cause the Resource Manager patch to pass an empty handle to _HandToHand, and calling _MemError will verify this.

What I recommend is to use _Get1Resource instead of _GetResource. The former does NOT have the patch code mentioned above. This is because the Dialog Manager doesn't call _Get1Resource and _Get1Resource doesn't call _GetResource. So, the Dialog Manager patch code containing _HandToHand will not be called if you use _Get1Resource.

This patch is contained in Macs with color and applies to the dctb and actb only. I believe this patch was introduced in System 6.0x.

Jim Reekes E.O., Macintosh Developer Technical Support

--
Mark B. Johnson AppleLink: mjohnsonDeveloper Technical Support domain: mjohnson@Apple.com

●●●

From: han@apple.COM (Byron Han, Project Scapegoat)
Subject: Re: Puzzling Resource Problem

In article <1765@thumper.bellcore.com> sdh@thumper.bellcore.com (Retief of the CDT) writes:

> On a reply.good,
> it tries to open the file with OpenResourceFile() (I believe that's the
> call I may have mistyped it here), sending in the filename from the
> standard file routine. It fails and returns fnfErr (file not found) no
> matter what.
>
> So what's the deal? How do I get a resource file opened from a name
> selected from a standard file dialog?

Well, you need to go the following (pseudo code)

```
err := GetVol(@fooName, savedVRefNum);
err := SetVol(NIL, theReply.vRefNum);

newResourceFile := OpenResFile(theReply.fName);

err := SetVol(NIL, savedVRefNum);
```

This is do guarantee that you open the file from the proper working directory. SFxxxFile returns in theReply.vRefNum the working directory reference number of the folder. The WD refnum is an integer that specifies both a volume reference number and a directory ID.

Hope this helps.

Byron Han, CommToolbox Scapegoat

●●●

From: han@apple.COM (Byron Han, Project Scapegoat)
Subject: Re: Interesting problem with GetResource('dctb', id)

In article <232@fwi.uva.nl> dolf@fwi.uva.nl (Dolf Starreveld) writes:

> I have written a general purpose "CopyResource" routine. The routine itself
> was extensively tested and seemed to work, but .....
> The copying of the 'dctb' resource was only recently added and things
> started to go wrong. After lots of debugging I found out that after executing
> the following statement:
> inHandle = GetResource('dctb', 10000);
> "inHandle" contains a valid handle, but not a handle to a resource. I entered
> TMON to find out more and:
> 1) Just before executing the statement the 'dctb' resource is not
> loaded in the heap.
> 2) Just after the statement, the 'dctb' resource is loaded, but
> immediately following it, the heap contains another relocatable
> block of exactly the same size of the resource (48 bytes).
> 3) The GetResource call returns a handle to this last block instead
> of to the actually loaded resource.

Well, this is kind of a strange quirk in the operating system.

Do this instead:

```
SetResLoad(FALSE);
hDCTB := GetResource('dctb', foobar);
SetResLoad(TRUE);
LoadResource(hDCTB);
```

This should work in place of a simple GetResource.

Byron Han, CommToolbox Scapegoat

●●●

From: alan@Apple.COM (Alan Mimms)
Subject: Re: Interesting problem with GetResource('dctb', id)

In article <232@fwi.uva.nl> dolf@fwi.uva.nl (Dolf Starreveld) writes:

> >started to go wrong. After lots of debugging I found out that after executing
> >the following statement:
> > inHandle = GetResource('dctb', 10000);
> >"inHandle" contains a valid handle, but not a handle to a resource. I entered
> >TMON to find out more and:
> >1) Just before executing the statement the 'dctb' resource is not
> > loaded in the heap.
> >2) Just after the statement, the 'dctb' resource is loaded, but
> > immediately following it, the heap contains another relocatable
> > block of exactly the same size of the resource (48 bytes).
> >3) The GetResource call returns a handle to this last block instead
> > of to the actually loaded resource.
> >
> >--dolf

SOME version of the system software introduced a "feature" in which 'dctb' and 'actb' resources are CLONED when they're retrieved by GetResource and related calls. A colleague of mine discovered this recently. You MAY be able to get around this with either Get1Resource calls or with the sequence:

```
SetResLoad (false);
GetResource (...);
SetResLoad (true);
LoadResource (...);
```

Anyone in System Software care to comment?

Also, please note that you really SHOULD go ahead an upgrade to 6.0.4 final since there were a few bugs fixed in the last few versions of 6.0.4 after b15 (I THINK).

Hope this helps.
--

Alan Mimms Communications Product Development Group

●●●

From: Steve M. Hoffman
Subject: Re: Altering the Resouce fork

I've been doing that for years. I think I read somewhere that you should never change the resources of the running program, but I've NEVER had a problem.

What I do is create a resource called PREF for my preferences. ID can be anything but I start at 0. I do a GetResource("PREF",0) and if NIL is returned then you create a default resource and write it. Then when you quit your program, call ChangedResource followed by WriteResource. You can find all of this in IM vol. 1 chapter 5. As for the data type of a PREF. It's whatever you want:

```
myPrefHandle=^myPrefPtr;
myPrefPtr=^myPrefRec;
myPrefRec=RECORD
        .. your fields here ..
        end; (* record *)
```

It's really a neat thing, but do some checking after GetResource in case someone has been poking around with ResEdit where they shouldn't. If you get back an invalid PREF, just go back to the default.

Steve M. Hoffman

●●●

From: lsr@Apple.COM (Larry Rosenstein)
Subject: Re: OpenResfile Question

In article <13325@unix.SRI.COM> mxmora@unix.SRI.COM (Matt Mora) writes:

> Now for the problem. OpenResFile doesn't seem to work right. It seems
> like OpenResFile wants a full pathname.

Yes.

On an HFS system, you can also use OpenRFPerm, which takes a vRefnum (could be a working directory refnum) as well as a file name. If all you have is a dirID, you'll have to open a working directory yourself.

> I tried to set the directory with PBHSetVol and then openresfile but that

Offhand, I don't know why this didn't work. There is a pitfall with PBHSetVol, in that if some piece of code subsequently calls GetVol, it will get back the root of the volume and not the folder you set. Perhaps that's what's happening here. (You can verify this by checking whether you can open a file at the root level.)

You might also consider setting ResLoad to FALSE before opening the file, just in case the application has resources that are marked preload.

Larry Rosenstein, Apple Computer, Inc.

●●●

From: mxmora@unix.SRI.COM (Matthew Xavier Mora)
Subject: How to Openresfile [Summary]
Keywords: Openresfile,PBGetCatInfo,PBHSetVol,HOpenResfile

Thanks for all your help with my Openresfile question.
(Larry,Tom,John,Bill,Bruce and any others I forgot to mention)

What was I doing wrong? I wasn't using the correct dirId when I was using the PBHSETVOL function. On page 54 of the UMPG Tim Maroney writes:

> In article <37560@apple.Apple.COM> keith@Apple.COM (Keith Rollin) writes:
> >I think that what is happening is that your ioDirID field is being changed from
> >MyCurID on the PBGetCatInfo call. Then, when you try the PBSetCatInfo call,
> >you are using a DirID different from the original one.
>
> "Yes. PBGetCatInfo changes the dirID it returns, apparently to the file ID.(Yet
> another place where this pointless feature is a pain in the ass.) It is
> necessary to save the dirID before calling PBGetCatInfo, then to reset the
> ioDirID field to this saved value before calling PBSetCatInfo. And of course,
> this is completely undocumented, unless you count the mysterious two-headed

arrow before ioDirID in the description of the PBGetCatInfo routine."

My error was that I was using the DirID that I received back from PBGetCatinfo when I found a file that I wanted to open. I thought that the DirID was the Directory the file was in. Apparently this is not true. It must be the File ID or something. What you need to do is when you test to see if it a file or a folder,if it is a folder save that dir id and vrefnum so you can use it later. The next PBGetCatinfo call will likely change those values.

What I was doing: {in pseduo code}

```
Addnametolist(myCPB)
   myPB.dirid:=mCPB.ioDirID;  {<----WRONG}
   PBHSetVol(mypb, false);
   OpenResFile(name);
end;

enumeratecatroutine(dirid);
  PBGetCatInfo(mypb, aSync);
    if a folder then
       enumeratecatroutine(dirid);
    else
      if a_file_I_want Addnametolist(MyCPB);
    end;
 end;
```

What I should have been doing:

```
Addnametolist(myCPB)
   myPB.dirid:=CurDirID;      {curDirId is a global for the current dir}
   PBHSetVol(mypb, false);
   OpenResFile(name);
end;

enumeratecatroutine(dirid);
 PBGetCatInfo(mypb, aSync);
  if a folder then
   CurDirID=dirid
   CurVrefNum=mycbp.iovrefnum
   enumeratecatroutine(dirid);
  else
    if a_file_I_want Addnametolist(MyCPB);
  end
end
```

What I'm doing now:

```
Addnametolist(myCPB)
   SetResLoad(false); {make sure not to read in  Preload set resources }
                      {thanks Larry}
   HOpenResFile(curVrefNum, CurDirID, name, perm);
end;

enumeratecatroutine(dirid);
 PBGetCatInfo(mypb, aSync);
 if a folder then
    CurDirID=dirid
    CurVrefNum=mycbp.iovrefnum
    enumeratecatroutine(dirid);
 else
   if a_file_I_want Addnametolist(MyCPB);
 end
end
```

Apparently there are four ways to open a resource file.

1. Openresfile (fullpathname) Probably not the recommended way

2. PPBOpenWD(pb, aSync);
   OpenRFPerm(name, vRefNum, perm); {don't forget stripaddress on filename}

3. GetVol(OldVol, vRefNum);
   PBHSetVol(pb, aSync);
   OpenResFile(name);
   SetVol(OldVol, vRefNum);

4. HOpenResFile(curVrefNum, CurDirID, name, perm);

The last one is a new High-level call documented in tech note #218. It does the equivalent of number 3.

Now here is another question. Which one do you think would be the fastest?

Thanks again.

--
Matthew Mora | my Mac Matt_Mora@sri.com

●●●

From: oster@well.sf.ca.us (David Phillip Oster)
Subject: Re: OpenResfile Question

Inside Mac Vol 4 documents a variant of OpenResFile called something like OpenRFPerm() that takes a vrefNum. If you have a dirId, you can create a temporary wRefNum for it you can use (remember to use your own application signature, and to dispose it after you are done with it.)

-- David Phillip Oster

●●●

Chapter 13 Sound & the Sound Manager

Need help with the new sound manager! (with code) 208
Synchronous sounds using the Sound Manager 209
Synchronous sounds using the Sound Manager 209
How to get rid of clicking? 210
Audio Phone Dialer Software 211
Simultaneously played digitized sounds 212

From: stoms@castor.ncgia.ucsb.edu (David Stoms)
Subject: Re: Need help with the new sound manager! (with code)

In article <900004@hpvcfs1.HP.COM> stevem@hpvcfs1.HP.COM (Steve Miller) writes:

> >OK boys and girls, I'm about at my wits end on the subject of playing 'snd '
> >resources on the Mac. I'm trying to use snd's in a simple game where the
> >sounds are played asynchronously while arcade style animation is occuring.

I found this in one of my text directories (I can't verify that this is right)----

In article <1354@ndmath.UUCP> milo@ndmath.UUCP (Greg Corson) writes:

> >Would anyone happen to have a code fragment that illustrates how to setup
> >the required callback routines so you can have SoundPlay play a sound in the
> >background (async) mode?
> >
> >1. open the channel and specify a callback
> >2. play a snd resource
> >3. have the callback set a global flag
> >4. have my main event loop close the channel when the flag is set.

This is the code I am using. There is NO guarantee or warranty with this code. Use at your own risk! This is NOT Apple DTS sanctioned code.

1. the main application needs to keep a global variable is a SndChannelPtr. Initialize it to NIL.

2. Pass in the handle to the snd resource that you want to play.

This should bootstrap to you greater understanding of the world of the Sound Manager.

```
PROCEDURE PlayIt(VAR chan: SndChannelPtr; theSound: Handle);
VAR
    err: OSErr;
    myWish: SndCommand;

BEGIN
    IF chan <> NIL THEN   {cancel sound in progress}
    BEGIN
        err := SndDisposeChannel(chan, TRUE);  {TRUE means now}
        IF err <> noErr THEN
        BEGIN
            DebugStr('Cannot dispose of old channel');
            Exit(PlayIt);
        END
        ELSE
            chan := NIL;
    END;

    err := SndNewChannel(chan, 0, 0, NIL);
    IF err <> noErr THEN
    BEGIN
        DebugStr('Cannot allocate new channel');
        Exit(PlayIt);
    END;

    IF theSound <> NIL THEN {we have a handle}
        IF theSound^ <> NIL THEN {and is not a nil handle }
        BEGIN
            err:=SndPlay(chan, theSound, TRUE);
            IF err <> noErr THEN
            BEGIN
                DebugStr('Cannot play sound');
                Exit(PlayIt);
            END;
```

```
            WITH myWish DO   { set up a sound mgr command block }
            BEGIN
               cmd := freeCmd;  {to free self when done }
               param1 := 0;
               param2 := 0;
            END; {with}

            err := SndDoCommand(chan, myWish, FALSE);

            IF err <> noErr THEN
            BEGIN
                DebugStr('cannot add command');
                Exit(PlayIt);
            END;
        END;
END;
```

●●●

From: lsr@Apple.COM (Larry Rosenstein)
Subject: Re: Synchronous sounds using the Sound Manager

In article <30976@ut-emx.UUCP> wras@walt.cc.utexas.edu (Steve Mariotti) writes:

> err = SndNewChannel(myChanPtr, sampledSynth, initMono, NULL);
> if (!err) {
> err = SndPlay(myChanPtr, soundHandle1, FALSE);
> err = SndPlay(myChanPtr, soundHandle2, FALSE);
>}

First, to play a sound while your programs continue, you need to pass TRUE for the last parameter. This specifies asynchronous playing, which is what you want.

Also, you can't (in general) play 2 snd resources with the same channel. When you play a resource, the synthesizer required by the resource is linked to the channel. If you link a syntehsizer more than once, you will (at best) not get any sound, and will probably crash.

For this reason, it is best to dispose of a channel after the sound is finished playing. To do this pass a call back routine when you create the channel, and send a callBack command after the SndPlay.

For the same reason, you should pass 0 instead of sampledSynth when creating the channel, in case the resource you play wants to use a different synthesizer.

If you know the structure of the resources you are going to play (ie, they have no synthesizer information) then you can play several sounds on the same channel. Or you can parse the sound resource to extract the actual samples and play them.

All this information comes from the revised Sound Manager documentation, which I think is available on Apple.COM (in /pub/dts/mac/docs). I've also got sample MPW Pascal code which implements an asynchronous SndPlay call.

Larry Rosenstein, Apple Computer, Inc.

●●●

From: cak3g@astsun9.astro.Virginia.EDU (Colin Klipsch)
Subject: Re: Synchronous sounds using the Sound Manager
Summary: Careful with SndNewChannel!

In article <8548@goofy.Apple.COM> lsr@Apple.COM (Larry Rosenstein) writes:

>In article <30976@ut-emx.UUCP> wras@walt.cc.utexas.edu (Steve Mariotti)
>writes:
>> err = SndNewChannel(myChanPtr, sampledSynth, initMono, NULL);

>> if (!err) {
>> err = SndPlay(myChanPtr, soundHandle1, FALSE);
>> err = SndPlay(myChanPtr, soundHandle2, FALSE);
>>}

A minor point that I missed on first reading of the Sound Manager chapter: the first parameter is a VARIABLE sound channel pointer; i.e. the calling definition of SndNewChannel goes like:

```
function SndNewChannel (VAR: chanPtr; ...)
```

In assembly language this requires extra care. You must be sure that you:

a) set your chanPtr to NULL or to the pointer of your own created pointer-block (using _NewPtr, size 1060 bytes);
b) pass the ADDRESS of your chanPtr to _SndNewChannel, NOT the pointer itself.

So it goes like this. . . (with the appropriate push/pop macros)

```
    clr.l  myChanPtr(A5)  ;if you're lazy; otherwise make your own
    push.w #0      ;space for the function result
    pea    myChanPtr(A5)  ;NOT push.l myChanPtr(A5)!
    push.w #0      ;let SndPlay link a synthesizer typically
    push.l #0      ;(is this a longint? my docs aren't handy)
    pea    MyCallBack     ;you definitely want this if you're going
               ; to be playing asynchronously
    _SndNewChannel
    pop.w  D0      ;error, if any
    ...            ;and on and on

MyCallBack     ; This routine will execute in response to a callBack
       ; command, which you pass to your channel with _SndDoCommand
       ; after you're finished with the channel.  This routine
       ; should set a flag somewhere indicating the channel
       ; is to be killed with _SndDisposeChannel as soon as is
       ; convenient, but this routine should NOT do the disposing
       ; itself, since it is not allowed to use the Memory Manager.
```

I realize the original poster was using Pascal, not assembly language, but I hope this can help anyway.

>Also, you can't (in general) play 2 snd resources with the same channel.
>When you play a resource, the synthesizer required by the resource is
>linked to the channel. If you link a syntehsizer more than once, you will
>(at best) not get any sound, and will probably crash.

Precisely, although I have never understood why the Sound Manager works in this way. Why can't SndPlay check to see if a particular synthesizer has already been linked to this channel, and if so, don't try relinking? SndPlay could be much more robust, it seems to me, but then again I wouldn't want to have to write it. . .

Colin Klipsch

●●●

From: oster@well.sf.ca.us (David Phillip Oster)
Subject: Re: How to get rid of clicking?
Keywords: Sound Driver

In article <1990Jun4.020601.9644@msuinfo.cl.msu.edu> gurney@cpsin2.uucp (Eddy J Gurney) writes:

>Also, while I'm posting... has anybody come up with a way to specify a
>FREQUENCY to FTSoundRec.sound1Rate instead of the rate? IM II gives the
>equation:
> frequency = FixMul(rate,FixRatio(22257,256))
>I solved this for frequency, and got: (Actually, I had my HP-48SX do it. I
>didn't want to be flamed for not solving correctly :-)
> rate = FixRatio(frequency,FixRatio(22257,256))

in floating point, this is:

```
    rate = frequency / (22257. / 256.);
```
Or, in fixed point:
```
    rate = FixDiv(FixRatio(frequency, 1), FixRatio(22257,256))
```

If the FixRatio(frequency, 1) offends you , you can also say:

```
#define FIXED(f) ( ((long) (f)) << 16)
and say    FIXED(frequency)
```

--
-- David Phillip Oster -

●●●

From: oster@well.sf.ca.us (David Phillip Oster)
Subject: Re: Audio Phone Dialer Software

In article <3397@ssc-vax.UUCP> housen@ssc-vax.UUCP (Kevin Housen) writes:

> >he needs some software (preferably a DA) to generate
> >appropriate dial tones. This would be pretty easy to
> >write, but I wanted to check first to see if there are
> >any PD/shareware DAs that will do the trick.

Both hypercard and Address Book Plus (a commercial product from Power Up Corporation, that I wrote) dial the phone by generating audio tones. AB+ uses a d.a., but it is commerical software, not shareware or public domain.

You can just hold the reciever up to the Mac on later Macs, but my testing shows that not all raw MacPluss have enough volume to generate all the tones reliably enough for the phone system. Experiment.

To dial the phone, you need two simultaneous tones. Hypercard and I use the Sound Driver's 4 voice synthesizer to produce two pairs of tones by doubling up the voices. Apple Developer Tech Support has been saying for some time that they want people to stop using Sound Driver and do everything with Sound Manager. Unfortunately, at least as of System 6.0.4, 4 voice sound was not supported by Sound manager on Macpluss. Here are the numbers from my dialer:

```
typedef struct Freq2 {
    Fixed a, b;
}Freq2;

/* These are the frequencies for the touch tone phone pad.
    Source: The TV Typewriter Cookbook, Don Lancaster, p.178
    Units for the table in comments is Hertz.
    The actual table was generated by applying the formula:
    hz = 1000000 / ( 44.93 * 256  / rate )
    to the table to get a new table I can use at interrupt time.
    This table uses the same values hypercard does.
 */
static Freq2 freqs[] = {
    { 0x000F5DDE, 0x000AD2C9},      /* { 1336, 941},      /* 0 */
    { 0x000DE7EA, 0x00080453},      /* { 1209, 697},      /* 1 */
    { 0x000F5DDE, 0x00080453},      /* { 1336, 697},      /* 2 */
    { 0x0010FD06, 0x00080453},      /* { 1477, 697},      /* 3 */
    { 0x000DE7EA, 0x0008DB46},      /* { 1209, 770},      /* 4 */
    { 0x000F5DDE, 0x0008DB46},      /* { 1336, 770},      /* 5 */
    { 0x0010FD06, 0x0008DB46},      /* { 1477, 770},      /* 6 */
    { 0x000DE7EA, 0x0009CCB9},      /* { 1209, 852},      /* 7 */
    { 0x000F5DDE, 0x0009CCB9},      /* { 1336, 852},      /* 8 */
    { 0x0010FD06, 0x0009CCB9},      /* { 1477, 852},      /* 9 */
    { 0x0012C9B2, 0x00080453},      /* { 1633, 697},      /* A */
    { 0x0010FD06, 0x0008DB46},      /* { 1477, 770},      /* B */
    { 0x0010FD06, 0x0009CCB9},      /* { 1477, 852},      /* C */
    { 0x0010FD06, 0x000AD2C9},      /* { 1477, 941},      /* D */
    { 0x000DE7EA, 0x000AD2C9},      /* { 1209, 941},      /* * */
    { 0x0010FD06, 0x000AD2C9}       /* { 1477, 941},      /* # */
};
```

--
-- David Phillip Oster -

●●●

From: jmunkki@hila.hut.fi (Juri Munkki)
Subject: Re: Simultaneously played digitized sounds
Keywords: divide

In <25974@pasteur.Berkeley.EDU> eta@ic.Berkeley.EDU.UUCP (Eric T. Anderson):

> >Hey -- be careful. You know you're throwing away bits for no reason
> >if you pre-divide. How about add first and then post-divide? On a
> >Mac, doing byte-addition is just as quick as 16-bit addition (except
> >if you're adding constants). Right?

Every cycle counts when you are doing things like this. I found it optimal to pre-divide. I also used only 11 Khz sound. Doing two 11 Khz sounds withpre-divide doesn't add all the much overhead, so my animation routine were not hurt too badly. Two sound channels is infinitely better than just one.

Juri Munkki Macintosh Support jmunkki@hut.fi PS /

Chapter 14 Windows & Grafports

Checking the validity of a refCon (window refcon) 214
Floating Windows 215
How do you find a window's location? 215
How do you find a window's location? 216
How do you find a window's location? 216

From:Tim Maroney
Subject: Re: Checking the validity of a refCon (window refcon)

In article <17662@dartvax.Dartmouth.EDU> ari@eleazar.dartmouth.edu (Ari Halberstadt) writes:

> >Here's a question for you folks out on NetLand. I'm trying
> >to figure out, with a reasonable degree of success, if the refCon
> >field of a window contains a valid handle. This must be determined
> >by a driver, which is separate from the the program which actually
> >created the window.

But it *is* your program, right? All the windows whose refCons you wil be looking for (if not at) were created by an application you wrote? Then probably the best check involves a little-known feature of the WindowRecord. You can put any value of 8 or greater into your own application's windows' windowKind field (IM I-276). Pick some arbitrary large number and the collisions with other applications using the same trick become unlikely. If your first check is this, then you should still do some other checks, but they're really just formalities. You will almost certainly never find another application which both uses this obscure trick and picks the same number for its windowKinds.

I'm kind of confused about when it is you expect to be looking at other windows. You should be able to check whether it's your application running in a variety of ways. You can check for the presence of your signature resource; you can have the application call your driver to tell it when the application is running, or even when it creates or destroys a window; you can install a 'STR ' with a particular ID that contains some distinctive string; and so on.

> >The tests currently performed are:
> >
> >Boolean ValidHandle(Handle hndl)
> >{
> > hndl = (Handle) GetWRefCon(window); /* copied here for brevity */
> > if (! hndl)
> > return(false);
> > if ((long) hndl & 1L) /* check for odd address */
> > return(false);
> > if (GetHandleSize(hndl) != expected_size)
> > return(false);
> >}

This looks mostly OK; it wouldn't be a bad idea to combine this kind of testing with one of the other kinds I mentioned. I do have a problem with the GetHandleSize. It could conceivably be called on a non-handle; the "hndl & 1" test doesn't establish that it's a handle. For instance, suppose the window belongs to an application that sticks the number 0xfffffffe into the refCon. Boom. You should check that the handle is even, then that it is a pointer into the application heap, then that what it points at is also an even pointer into the application heap, and then dereference the pointer to find out whether the longword at it is equal to some distinctive value you've stashed in all your refcon records at the beginning.

> >Actually, I've got three questions.
> >1. Is it ok to check for odd adresses on a 680x0? I assume Apple
> >wouldn't write NewHandle to create something on an odd address, since
> >access times are slower.

NewHandle will always create it on an even address, regardless of machine. Heaps are divided into blocks which are size-rounded up to even numbers (IM II-25).

> >2. I would like to check that the handle is within a reasonable area
> >of memory, i.e., within the application heap. However, since the
> >application may have several heap zones, what is the correct way to
> >check this? Is it sufficient to use ApplicZone and GetApplLimit?

Sure. Be sure to convert them to unsigned longwords and call StripAddress on them before you do your arithmetic. The same goes for the pointers you're checking. Unless you do the (in my opinion, dreadful) trick of putting a heap on the stack, all your heaps will fall within the application heap. (You could also put a heap in temporary memory, but then it's not really temporary, is it?)

> >3. Any more ideas for useful checks? I've already got a special
> >long int stuck at the head of the place where the correct handle

>would point to, and I've thought of making sure the handle is
>below RomBase.

This shouldn't hurt, but it's redundant if you're already checking against the application heap.

>--
>Ari Halberstadt '91, "Long live succinct signatures"
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: pepke@loligo (Eric Pepke)
Subject: Re: Floating Windows

In article <18786@bellcore.bellcore.com> sdh@flash.UUCP (Stephen D Hawley) writes:

>I need a pointer for making windows that float on top of the rest of the
>world no matter what. I tried setting GostWindow to the window I wanted
>on top, but that doesn't work. I want the kind of windows in FullPaint.
>
>Do I have to dick around with the window list pulling them in front
>constantly?

April 1988 MacTutor had a bit on this. I wouldn't do it exactly the way they did (and, in fact, I didn't) because, for one thing, Apple people have suggested that it would probably break. Most of the reason for this seems to have to do with accessing the Window Manager port directly to change the region where the drag window rectangle shows.

There is no real trick, just a lot of special cases. Don't use GhostWindow. The first order solution is to keep a floating window on the top, put it on the top and never allow another window to come in front of it. This is relatively easy; don't use SelectWindow on a document window, but rather activate the window and bring it to just behind the rearmost floating window. The trickiness comes with desk accessory windows. You want the floating windows to be in front of all the application windows, but not in front of the desk accessory windows.

I found that the easiest way to solve the problem was to think about all the special cases in the context of the structure of my main loop. What should happen when you bring up a desk accessory in front of the floating windows? What then should happen when you bring up a document window in front of that? Do you disallow that, as HyperCard does, or do you do it somewhat more cleverly? What window should be highlighted as active? (HyperCard does it
stupidly.)

Eric Pepke INTERNET: pepke@gw.scri.fsu.edu

●●●

From: murat@farcomp.UUCP (Murat Konar)
Subject: Re: How do you find a window's location?

In article <3258@usceast.UUCP> jwwalker@usceast.UUCP (Jim Walker) writes:

>I know how to set a window's location with MoveWindow and
>SizeWindow, but how do you *find* a window's location? If the
>window is visible, I can use
> (**((WindowPeek)window_ptr)->contRgn).rgnBBox
>but that seems to contain garbage when the window is invisible.
>Yet the Window Manager somehow keeps track of the locations of
>invisible windows.

Here's how I do it:
1) Save the ccurrent port
2) SetPort to the window in question
3) Grab a copy of its port rect
4) Do a LocalToGlobal on its topLeft and botRight
5) Restore the current port

VOILA!

You now have the window's location in global coordinates!

Murat N. Konar

•••

From: wdh@well.sf.ca.us (Bill Hofmann)
Subject: Re: How do you find a window's location?
Summary: GlobalToLocal the topLeft of the portRect

In article <22352@dartvax.Dartmouth.EDU> mjm@eleazar.dartmouth.edu (Michael McClennen) writes:

> >jwwalker@usceast.UUCP (Jim Walker) writes:
> >>I know how to set a window's location with MoveWindow and
> >>SizeWindow, but how do you *find* a window's location? If the
> >>window is visible, I can use
> >> (**((WindowPeek)window_ptr)->contRgn).rgnBBox
> >>but that seems to contain garbage when the window is invisible.
> >>Yet the Window Manager somehow keeps track of the locations of
> >>invisible windows.
> >How about checking the grafPort's portRect?

The portRect of a grafPort is the top,left,bottom,right of the window in LOCAL coordinates, that is, the coordinate system of the window. (Note that the top,left need not be 0,0). What you want is the position of the top,left in GLOBAL coordinates. So, looking at Inside Mac Vol I-193:

```
Point getWindowLocation(WindowPtr wp)
{
    Point   where;
    GrafPtr    curPort;

    where.v = wp->portRect.top;
    where.h = wp->portRect.left;
    GetPort(&curPort);
    SetPort(wp);
    GlobalToLocal(&where);
    SetPort(curPort);
    return(where);
}
```

If you want to include the title bar, which is above and to the left of the content, you'll have to look at the strucRgn of the window.

-Bill Hofmann

•••

From: amanda@mermaid.intercon.com (Amanda Walker)
Subject: Re: How do you find a window's location?

> > jwwalker@usceast.UUCP (Jim Walker) writes:
> > >how do you *find* a window's location?

In article <22352@dartvax.Dartmouth.EDU>, mjm@eleazar.dartmouth.edu (Michael McClennen) writes:
> > How about checking the grafPort's portRect?

The portRect is in local coordinates. What I do is SetPort to the Window, make a copy of the portRect, and call LocalToGlobal on the corners. If you really want to be studly, you should also save & restore the zoomed state...

--
Amanda Walker

•••

Chapter 15 Articles & Notes

Macintosh One-Liners 221
Scheme to Manage a "Windows" menu 236
How to write an INIT in Pascal 241
How do you play Asynchronous Sound? 243
Default 2.1 CDEF 246
ToolBox Gotchas 254
How do you hide the menu bar? 256
BitMap Rotation in C (and support routines) 259
INIT Skeleton Code 268
New Volume Scanning Algorithm 275

Macintosh One-Liners
by Eric Pepke

The Macintosh One-Liners are intended to condense into a small space information about some of the most common Macintosh problems and programming pitfalls. Each one-liner is a single line of text, shorter than 80 characters, which informs about one aspect of Macintosh use or programming.

One-liners give either facts or advice. The facts may be obvious to some people and obscure to others but are important for all. The advice is intended to help keep people from running into the most common nontrivial problems. Like proverbs, the advice may not be absolute and may sometimes be more conservative than is strictly necessary. However, I have found that a little constructive paranoia can go a long way toward avoiding problems, and more than once I have taken a precaution which seemed extreme at the time but which saved my skin later on.

The one-liners started as a list I made for myself of things to remember while writing programs. I have augmented them with my condensed records of several years of Info-Mac, Usenet, and Delphi digests and one year of Usenet reading. People who have contributed to the list since its first release are mentioned at the end. The result is very much a gestalt of the Macintosh lore I have seen and depends on the wisdom and efforts of many people. If I have forgotten to include your name, I apologize.

The one-liners are organized into two sections. The first lists the one-liners under broad categories without explanation. This is good for cutting out and sticking on the wall. The second section gives some extra information on each one-liner.

All references given are to Apple documentation. In some cases, the references explain the problem, but in most, they just describe the parts of the Toolbox you need to use. Most of the problem fixes are a result of my experience and the experience of those people who are listed below. The acronym IM means Inside Macintosh; the page numbers refer to the Addison-Wesley trade paperback version. TN means Technical Note. I use the HyperCard stack, available from sumex-aim.stanford.edu, apple.com, and no doubt other places as well. I have avoided referring to any third-party software tools and only refer to a minimal set of Apple tools. This is not to endorse Apple in any way. It's just that I don't want to get into the product comparison business. I use a minimal set of ubiquitous tools, and I describe fixes in terms of those tools. If you have different tools, you can translate. I have also tried to avoid using language-specific statements where possible, but all my example code is in C. Again, you can translate.

Compiled by Eric Pepke
Additional material by Steve Maker, Keith Rollin, Gregory Dudek, Brian Bechtel, Henry Minsky, Carl C. Hewitt, Jim Lyons, Alex Lau, Kent Borg, Peter W. Poorman, Ross Yahnke, Mark Fleming, Mark Anderson.

Send suggestions to pepke@gw.scri.fsu.edu on the Internet or PEPKE@FSU on BITNET. Have fun.

The Main Loop and Events
Call MaxApplZone and MoreMasters when the application starts up.
If you call SetApplLimit, do it before calling MaxApplZone.
Use HT in MacsBug while running to estimate how many times to call MoreMasters.
Don't use SetEventMask to disable mouseUp events. Better not to use it at all.
SetPort to a known good grafPort once every time through the event loop.
Calling WaitNextEvent with more than 50 ticks will fail on some systems.
Set the cursor on suspend and resume events.
Call GetDblTime to get the maximum time for a double click.
Measure double click time from mouse up to mouse down.
Call either WaitNextEvent or both GetNextEvent and SystemTask.
Call IsDialogEvents and DialogSelect even if GetNextEvent returns false.

Menus
Use SetItem to include meta characters literally in menus.
Never make MENU resources purgeable.

Resources
GetResource never produces resNotFound. Check for a NIL handle instead.
To create a resource file, call Create, then CreateResFile.
To open a resource file read-only for shared access, use OpenRFPerm.
Don't leave ResLoad set to false.
GetResource on a dctb may return a non-resource copy of the dctb.

Windows, Alerts, and Dialogs
Drag windows to the bounding box of GetGrayRgn.
Hide scroll bars when deactivating a window.
Call DrawGrowIcon when activating or deactivating a window with a grow region.
DrawGrowIcon does not check to see if the window has a grow region.
Call PenNormal before calling DrawGrowIcon.
itemHit will not be set when a dialog filter is called.
Use a disabled UserItem to draw the roundrect outline around the OK button.
ModalDialog assumes the dialog is already visible and in the front.
Use screenBits . bounds to center dialogs, alerts, etc. below the menu bar.
If you save window locations in files, they may not be valid for all monitors.
DragWindow expects startPt in boundsRect; if not it may not call SelectWindow.
SelectWindow does not automatically call SetPort. You must do that yourself.
DialogSelect responds to activate events but ignores suspend/resume events.
Call PenNormal before calling DrawControls.
The Control Manager only works when the origin of a window is at (0, 0)
To find the position of a window, use LocalToGlobal on the origin.

Drawing
Always set the VisRgn and ClipRgn of offscreen ports.
Set the ClipRgn first when making a picture.
Don't make rowBytes in bitMaps greater than 8191.
To dim text, draw a rectangle with penPat=gray and penMode=patBic over it.
To draw rotated text, draw to an offscreen bitmap, rotate it, and CopyBits it.
Don't use picSize to determine the size of a picture. Check the handle size.
Never draw outside a window except in XOR mode for temporary effects like drag.
To avoid animation flicker, synchronize drawing to the vertical retrace.
Lock handles to pictures before calling DrawPicture.
CopyBits on more than 3Kb data will work, but it might have to allocate memory.
The small Mac screen is 512 pixels wide by 342 pixels high including menu bar.

Interrupts and VBL Tasks
Don't call any Memory Manager routines during an interrupt.
Unlocked handles may not be valid during an interrupt.
To synchronize to the vertical retrace on Macs with slots, use SlotVInstall.

Files
Don't write in the application file. This will fail with read-only devices.
Use PBGetVInfo to convert a VRefNum to a volume name.
Delete uses the Poor Man's Search Path, so don't delete blindly.
File Manager routines with dirID=0 use the Poor Man's Search Path.
Truncate and reallocate files before overwriting to reduce fragmentation.
Check/change the creator and type of Save As... files before overwriting.
If you rewrite files by deleting and creating, copy all Finder information.
Directory ID's are longs, not shorts. Shorts work ALMOST all the time.
If a file version number appears in a file manager call, always set it to 0.
To convert a pathRefNum to a name or file number, use PBGetFCBInfo.
Prevent the creation of files with names that begin with a period.
Write/update the Finder info before writing data or resource forks.

Handles and Pointers
Lock handles before passing their dereferenced pointers to any routine.
Lock handles before setting referenced data to expressions containing functions
Put an odd long at location zero on a 68000 to help find NIL handle references.
Call MoveHHi before locking a handle to avoid memory fragmentation.
Use HGetState/HLock/HSetState to lock a handle then put it back as it was.

General
Always use unsigned characters within text and Pascal-format strings.
Save application preferences in a folder named Preferences in the System Folder
Use SysEnvirons to find the System (Blessed) Folder.
Use GetAppParms to get the name of the application.
The high bit of SysParam . volClik enables the alarm clock.
Check the application name at $910 before exiting with ES within MacsBug.
To exit to shell in the mini-debugger, enter SM 0 A9 F4 and then G 0.
In Pascal, don't nest procedures to be passed by procedure pointer.
In Pascal, "with theHandle^^" is unsafe if memory compaction can occur.

Controversy Corner (Don't shoot me; I'm just the messenger.)
Avoid writing tail patches for traps.
There is no official way to tell if MultiFinder is running or not.

[*This section was created by Eric after my futile attempt at trying to explain his one-liners. I showed him my idea and he was real interested and expanded on it. The idea was that his one-liners was very helpful at remembering the correct things to do but to help some beginning programmers , I thought that an explanation would help them understand why. Also I added a few comments and they are shown by square brackets. MXM*]

Call MaxApplZone and MoreMasters when the application starts up.
References: IM II-30, IM II-31, TN 53

A call to MaxApplZone is needed for an application to get as large a heap as it can under all systems. Usually, this is what you want to do. Call MoreMasters early in the application to get enough blocks of master pointers for the anticipated number of relocatable memory blocks to be allocated. [You should also call MoreMasters from code segment 1 and not from your intialization segment.]

If you call SetApplLimit, do it before calling MaxApplZone.
References: IM II-30

MaxApplZone expands the application zone to the heap limit set by SetApplLimit. It doesn't make any sense to call MaxApplZone first. Most applications will not need SetApplLimit at all, but the few that do should call it before MaxApplZone.

Use HT in MacsBug while running to estimate how many times to call MoreMasters.
References: IM II-22, IM II-31, MacsBug 6.0, TN 53

There is an art to esimating how many times to call MoreMasters. If you call it too few times, the Memory Manager will be forced to call it later on in the application, which can fragment the heap. If you call it too many times, you will be wasting master pointers that you do not need. It's usually better to call MoreMasters too many times than too few. To estimate the number of times, you need to have some idea of how many relocatable blocks your application typically allocates. A good way of finding out is to run the application for a while, exercising all its features. Then find out how many relocatable memory blocks have been allocated. The HT command in MacsBug will provide this information. Once you have this number, divide by 64, add some slop (maybe 25%), and that's the number of times you need to call MoreMasters.

Don't use SetEventMask to disable mouseUp events. Better not to use it at all.
References: IM II-70, TN 202

SetEventMask should only be used to enable KeyUp events if an application needs them. Some programmers, especially in the early days, used SetEventMask to disable MouseUp events, which caused a lot of problems. One particularly strange one is that, after running them, double-clicking would no longer work in the Finder. Because the Finder could not see the MouseUp, it could not detect a double-click.

SetPort to a known good GrafPort once every time through the event loop.
References: IM I-165

You never know if a desk accessory is playing by the rules. Even if you do things absolutely correctly in your application, there is a bizarre set of circumstances which can (and has) resulted in crashes. Say desk accessory A plays hard and fast with the current GrafPort. Say it has the current GrafPort set to a window, and it closes the window. The current GrafPort becomes invalid. Now, desk accessory B does something which requires the current GrafPort to be valid, but does not explicitly set it first. This is common because there are calls that expect some GrafPort to be current, but don't really care which one, and some programmers are sloppy. When B tries this, kabloom! Even though your application is perfectly innocent, it will probably be blamed anyway. If you do a SetPort to a known good GrafPort every time through your event loop, this will not happen.

Calling WaitNextEvent with more than 50 ticks will fail on some systems.
References: TN 177

MultiFinder 1.0 had a bug which caused it to hang if you called WaitNextEvent from the background with more than about 50 ticks. The bug has since been fixed, but you never know what weird system will be used to run your application.

Set the cursor on suspend and resume events.
References: IM I-167, Programmer's Guide to MultiFinder

When you receive a resume event, another application may have changed the cursor, so set it to whatever you need it to be, even if it is just an arrow. When you receive a suspend event, set the cursor to an arrow as a courtesy to other applications whose programmers are not as well informed as you.

Call GetDblTime to get the maximum time for a double click.
References: IM I-261

The double-click time can be set by the user via the Control Panel. Use this value to determine whether two sequential clicks formed a double click. People have different capabilities and natural speeds, and some users may be physically unable to make a double click at a certain speed. Also remember that the value of GetDblTime can change at any time, because the user can pull down the Control Panel at any time.

Measure double-click time from mouse up to mouse down.

The Finder measures a double click from a MouseUp event to a MouseDown event. This seems to be a very natural way of doing it.

Call either WaitNextEvent or both GetNextEvent and SystemTask.
References: Programmer's Guide to Multifinder

WaitNextEvent is not just a direct replacement for GetNextEvent. It performs the function of SystemTask as well.

[Example Below is from Apple's TESample Source Code]

```
procedure EventLoop;

{Get events forever, and handle them by calling DoEvent.}
{ Also call AdjustCursor each time through the loop.}

       var
           cursorRgn: RgnHandle;
           gotEvent: BOOLEAN;
           event: EventRecord;
           mouse: Point;

   begin
       cursorRgn := NewRgn;         {we'll pass an empty region to WNE the first time thru}
       repeat
           if gHasWaitNextEvent then
               begin  {put us 'asleep' forever under MultiFinder}
                   GetGlobalMouse(mouse);          {since we might go to sleep}
                   AdjustCursor(mouse, cursorRgn);
                   gotEvent := WaitNextEvent(everyEvent, event, GetSleep, cursorRgn);
               end
           else
               begin
                   SystemTask;                     {must be called if using GetNextEvent}
                   gotEvent := GetNextEvent(everyEvent, event);
               end;
           if gotEvent then
               begin
```

```
                AdjustCursor(event.where, cursorRgn); {make sure we have the right cursor}
                DoEvent(event);                        {Handle the event only if for us.}
            end
        else
            DoIdle;
    {If you are using modeless dialogs that have editText items,}
    {you will want to call IsDialogEvent to give the caret a chance}
    {to blink, even if WNE/GNE returned FALSE. However, check FrontWindow}
    {for a non-NIL value before calling IsDialogEvent.}
    until AllDone
end; {EventLoop}
```

Call IsDialogEvents and DialogSelect even if GetNextEvent returns false.
References: IM I-416

In addition to handling dialog events, these routines also control cursor flashing in dialog EditText items. They need to be called even as a result of null events to make sure the flashing occurs.

Use SetItem to include meta characters literally in menus.
References: IM I-346, IM I-357

AppendMenu recognizes certain characters as meta characters to control the text style, keyboard equivalents, and so on. If you want to display any of these meta characters within the text in the menu, you will need to use SetItem instead.

Never make MENU resources purgeable.

GetResource never produces resNotFound. Check for a NIL handle instead.
References: IM I-119

[quick example]

```
id:=1000;
myResHandle:=GetResource('ICON', id);
if myResHandle<>nil then
    PlotIcon(r, myResHandle);
else
    SysBeep(duration);
```

To create a resource file, call Create, then CreateResFile.
References: IM I-114, TN 101

CreateResFile will first check to see if the file exists, and if it does, it will add a resource fork to that file. CreateResFile uses the Poor Man's Search Path, so if you call it by itself, you may find that it actually modifies a file in the system folder rather than creating a file where you want it. To avoid this, call Create first to make sure a file exists where you want it.

To open a resource file read-only for shared access, use OpenRFPerm.
References: IM IV-17, TN 116, TN 185

Don't leave ResLoad set to false.
References: IM I-118

The SetResLoad routine should only be used in very special circumstances to prevent the automatic loading of resources into memory. Parts of the Toolbox require ResLoad to be true, so ResLoad should only be false for a very short time.

GetResource on a dctb may return a non-resource copy of the dctb.

When GetResource is called on a dctb, it may return a copy of the dctb rather than a handle to the resouce itself. This was done as a kludge to fix a bizarre problem. Of course, if you aren't aware of it, the kludge itself can cause some bizarre problems in programs that try to edit dctb resources.

Drag windows to the bounding box of GetGrayRgn.
References: IM I-282, IM I-289, IM V-208

DragWindow requires a bounds rectangle to specify the area over which the window can be dragged. Normally, it should be possible to drag a window anywhere. Once upon a time applications were recommended to use screenBits.bounds as the drag rectangle. Many pieces of Apple sample code recommend this to this day. However, on systems with multiple monitors, screenBits.bounds only enclose the screen with the menu bar. What you really want is to be able to drag the window within a rectangle that encloses all screens. The best one to use is the bounding box of the return value of GetGrayRgn. Because the GrayRgn covers all monitors, the bounding box is at least large enough. If the system is too old to have the GetGrayRgn function, you can use the GrayRgn global variable or just assume that screenBits.bounds is good enough. The following sample THINK C code drags theWindow around in response to an EventPtr called theEvent:

```
Rect boundsRect;
boundsRect = (*GetGrayRgn()) -> rgnBBox;
DragWindow(theWindow, theEvent -> where, &boundsRect);
```

Hide scroll bars when deactivating a window.
References: IM I-46

The entire scroll bar of an inactive window, including the arrows at the top and bottom, is supposed to be hidden. Few applications pay attention to this, but it is one of those little touches that really cleans up the feel of a program.

Call DrawGrowIcon when activating or deactivating a window with a grow region.
DrawGrowIcon does not check to see if the window has a grow region.
Call PenNormal before calling DrawGrowIcon.
References: IM I-287

DrawGrowIcon is not smart enough to know if the window has a grow region, so don't assume that you can always call it in a generic window handler. On the other hand, it is smart enough to know if the window is active or inactive, so it always does the right thing. It assumes certain things about the QuickDraw environment, such as a pen size of (1, 1), so it is safest to call PenNormal before every call to DrawGrowIcon.

itemHit will not be set when a dialog filter is called.
References: IM I-415, TN 112

This is one of the trickiest pitfalls to using dialogs. Dialog filters are nearly always used to provide interactions in userItems and are specific to those userItems. Thinking about that task, it is reasonable to assume that the filter can just test itemHit to see if the action is appropriate to the userItem. Though reasonable, the assumption is wrong. A dialog filter gets the raw event before any decisions are made. It is entirely responsible for determining where the click was and setting itemHit. An easy way to do this for a userItem is to do a PtInRect with the rectangle around the userItem. You can also use FindDItem.

[Sample below is from an old program that had a list manager in the dialog MXM].

```
function MyFilter (theDialog: DialogPtr; var theEvent: EventRecord; var itemHit: integer):
Boolean;

        var
            key: SignedByte;
            Itype: Integer;
            iBox, r: Rect;
            iHdl: Handle;
            MouseLoc: Point;
            isDbl: boolean;
            myList: ListHandle;
    begin
        Setport(theDialog);
        MyFilter := False;
        MyList := ListHandle(GetWRefCon(theDialog));
        MyList^^.port^.txSize := 9;
        MyList^^.port^.txfont := 4;
        case theEvent.what of
```

```
            keyDown, autoKey:
                begin
                    key := theEvent.message;
                    if key in [13, 3] then
                        begin
                            MyFilter := true;
                            itemHit := 9;            {Set Itemhit}
                        end;
                end;
            mouseDown:
                begin
                    mouseLoc := theEvent.where;
                    GlobalToLocal(mouseLoc);
                    GetDItem(theDialog, 3, iType, ihdl, ibox);
                    r := Ibox;
                    r.right := r.right + 16;
                    if PtInRect(mouseLoc, r) then
                        begin
                            isDbl := LClick(mouseLoc, theEvent.modifiers, Mylist);
                            MyFilter := true;
                            itemHit := 3;
                        end;
                end;
            UpdateEvt:
                begin
                    BeginUpdate(theDialog);
                    LUpdate(theDialog^.visrgn, Mylist);
                    MyList^^.port^.txSize := 9;
                    MyList^^.port^.txfont := 4;
                    DrawDialog(theDialog);
                    EndUpdate(theDialog);
                end;
        end;
    end;
```

Use a disabled UserItem to draw the roundrect outline around the OK button.
References: IM I-421, TN 34, default cdev

It seems strange that there is no trivial way to do such a basic feature of the Macintosh interface, but that's the truth. How does one make the roundrect outline around the default button that we all love so much? Just drawing it there after showing the dialog is not good enough, because if something happens to cover up the dialog, the outline will be erased. Some people use disabled PICT items containing roundrects, but you really have to make a custom PICT for every size of button to get the thickness of the line just right. A better way is to use a userItem that draws the outline. Add to the dialog a disabled userItem that is larger than the OK button by 4 pixels on each side. Make the dialog hidden, so that you will have to do a ShowWindow to show it. Then install a procedure for the userItem which draws the outline within its rectangle. Here are some THINK C code fragments:

```
pascal void DrawOKOutline(theDialog, whichItem)
DialogPtr theDialog;
INTEGER whichItem;
/*Draws an OK outline in a userItem*/
{
    INTEGER itemType;    /*The type of the item in GetDItem*/
    Handle theItem;   /*The handle to the item*/
    Rect itemBox; /*The box containing the item*/
    PenState savedPenState;  /*The old pen state saved and later restored*/

    GetDItem(theDialog, whichItem, &itemType, &theItem, &itemBox);
    GetPenState(&savedPenState);
    PenNormal();
    PenSize(3, 3);
    FrameRoundRect(&itemBox, 16, 16);
    SetPenState(&savedPenState);
}
```

```
    ...and when you want do use this userItem...

#define MyDlgRSRC whatever   /*Number of the dialog resource*/
#define MyDlgOutline whatever       /*The number of the userItem with the outline*/

    INTEGER itemType;    /*The type of the item in GetDItem*/
    Handle theItem;   /*The handle to the item*/
    Rect itemBox; /*The box containing the item*/

    /*Get the dialog box*/
    theDialog = GetNewDialog(MyDlgRSRC, (DialogPtr) 0, (WindowPtr) -1);

    /*Add the OK button outline*/
    GetDItem(theDialog, MyDlgOutline, &itemType, &theItem, &itemBox);
    SetDItem(theDialog, MyDlgOutline, itemType, &DrawOKOutline, &itemBox);

    /*Everything is set; show the dialog*/
    ShowWindow(theDialog);
```

If you want to get really fancy you can have the userItem procedure first look at the rectangle of the OK button, and then set its own rectangle accordingly.

[For an automatic way of doing this look for the Default cdef article by Lloyd Lim]

ModalDialog assumes the dialog is already visible and in the front.
References: IM I-415

A dialog has to be visible and above all other windows before you call ModalDialog. If you call ModalDialog without ensuring this, there will be big trouble.

Use screenBits.bounds to center dialogs, alerts, etc. below the menu bar.
References: IM I-163, IM I-527

screenBits.bounds is the bounding rectangle of the screen that has the menu bar. This is the most convenient place to put dialogs and alerts. With some simple calculation, knowing screenBits . bounds and the width and height of your window, you can center it. You can find out the width and height of the standard file dialogs by examining their DLOG templates.

If you save window locations in files, they may not be valid for all monitors.
Refernces: IM V-208

Saving the locations of windows between sessions is a nice thing for an application to do, but what if the user opens the document on a system with a very different screen setup? The window might not appear where it can be manipulated. So, if you do this, be sure to test to see if a window will appear in an accessible place on the screen, and if not, put it in a default location. You can do this by checking to see if a rectangle inset a few pixels within the title bar intersects the grayRgn.

DragWindow expects startPt in boundsRect; if not it may not call SelectWindow.
References: IM I-289

If the startPt you pass to DragWindow is not within the boundsRect, a click in the title bar may not call SelectWindow and bring the window to the front. Dragging will still work, but clicking in place won't. The easiest fix is just to make sure boundsRect is big enough. You can call SelectWindow yourself, but this affects the feel of window dragging.

SelectWindow does not automatically call SetPort. You must do that yourself.
References: IM I-284

Selection and drawing are independent of each other, although most of the time you want to do both. You can draw into windows that are not selected. QuickDraw does not set the port behind your back, which has the advantage of giving you flexibility. It has the disadvantage that you must remember to set the port yourself.

DialogSelect responds to activate events but ignores suspend/resume events.

References: IM I-417

If you get suspend and resume events, make sure to activate or deactivate dialogs on the screen in response to them.

Call PenNormal before calling DrawControls.
References: IM I-322

DrawControls assumes several things about the QuickDraw environment. Calling PenNormal is the best way to be sure that it does not mess up.

The Control Manager only works when the origin of a window is at (0, 0)
References: IM I-322

If you use SetOrigin to set the origin of a window, you must call SetOrigin(0, 0) before doing anything that can affect or draw controls.

To find the position of a window, use LocalToGlobal on the origin.
References: IM I-165, IM I-193

The safest way to find the position of a window is to do a SetPort to the window, get the origin of the window, and do a LocalToGlobal on that point.

Always set the VisRgn and ClipRgn of offscreen ports.
References: IM I-149

The assertion in Inside Macintosh that the VisRgn has no effect on images that are not on the screen is wrong. To be safe, always set both regions of offscreen GrafPorts.

Set the ClipRgn first when making a picture.
References: TN 59

The default ClipRgn for pictures is the largest possible region. If you create a picture using the default, as soon as you try to scale or move it using DrawPicture, the edges can overflow, causing all sorts of nasty things to happen. To avoid this, set the ClipRgn as the first thing when you create a picture.

Don't make rowBytes in bitMaps greater than 8191.
References: IM V-53

The top three bits of rowBytes are used by the first release of Color QuickDraw to keep information about the bitmap. More recent versions of Color Quickdraw only use the top two bits, but do you want to gamble that everybody in the world has upgraded? I thought not.

To dim text, draw a rectangle with penPat=gray and penMode=patBic over it.

Dim text is another of those things which is strangely absent from QuickDraw. You can dim text or anything else by just painting over it with a gray penPat in patBic mode.

To draw rotated text, draw to an offscreen bitmap, rotate it, and CopyBits it.

Although there are ways to rotate text at any angle in PostScript, QuickDraw is only able to draw text in one orientation. The only way to get rotated onscreen text is to rotate the bitmap. There is no call to do this; you have to do it yourself.

Don't use picSize to determine the size of a picture. Check the handle size.
References: IM V-87

The picSize field of a picture is only 16 bits wide. This is O.K. for most black-and-white pictures, but when Color QuickDraw was invented, it was discovered that this limitation was way too small. The

actual length of a picture is now determined by the size of the handle. Although picSize is maintained for compatibility in version 1 pictures, don't count on it.

Never draw outside a window except in XOR mode for temporary effects like drag.

Drawing outside a window is very dangerous. For one thing, historically, most programs which have done this broke when extensions were made to QuickDraw. For another thing, in most cases, the user does not expect anything except the Finder to draw on the desktop. If you need to draw in a window the size of the screen, why not open a window the size of the screen and use that? In general, only DragGrayRgn and XOR lines for zooms should be done outside a window. These are temporary effects which always go away, so they are not so bad.

To avoid animation flicker, synchronize drawing to the vertical retrace.
References: IM II-350, IM V-567

The video hardware is constantly sweeping through the screen memory displaying the bytes on the screen. If you try to access some memory during a QuickDraw operation that is currently being swept, there will be a conflict. Usually, QuickDraw is so fast that you will not notice it, but if you are doing animation using CopyBits, you might notice some flicker. You can try to avoid this by synchronizing the beginning of the CopyBits to the vertical retrace. The idea is always to have the drawing occur just ahead of the sweep so that no conflict occurs. A good way of finding out when the vertical retrace occurs is to install a VBL task on a small Macintosh, or use SlotVInstall on a Macintosh with slots. The routine will be called as a result of the vertical retrace, so it can signal another portion of the program to begin drawing. You will have to experiment with timing for your particular application.

Lock handles to pictures before calling DrawPicture.
References: IM I-190

Rumor has it there is a bug in DrawPicture which involves following a pointer to the picture data rather than a handle and an offset. As DrawPicture can at times cause a memory compaction, this is unsafe. I find this bug hard to believe, as it would be a monstrous oversight, but better safe than sorry. Lock the picture.

CopyBits on more than 3Kb data will work, but it might have to allocate memory.
References: IM I-188

On old systems, CopyBits had a problem with copying a picture bigger than about 3Kb. This was because CopyBits uses temporary memory from the stack. The bug required some programs to break down CopyBits calls into a number of thin wide strips, which was not very convenient. The bug has for a long time been fixed. What happens now is that CopyBits will first try to get enough memory from the stack and, if that is not enough, will allocate blocks using Memory Manager calls.

The small Mac screen is 512 pixels wide by 342 pixels high including menu bar.
Count 'em.

Don't call any Memory Manager routines during an interrupt.
Unlocked handles may not be valid during an interrupt.
References: IM II-195

An interrupt may interrupt anything, including a memory compaction. During a memory compaction, parts of the heap may be in an indeterminate state. Calling any Memory Manager routines could destroy the heap. Also, it is unsafe to look at unlocked handles. The memory in the handle may have been in the process of moving when the interrupt occurred.

To synchronize to the vertical retrace on Macs with slots, use SlotVInstall.
References: IM V-567

Don't write in the application file. This will fail with read-only devices.
References: TN-115,TN-116

You never know where the file containing your application resides. It could be on a CD-ROM, or it could be on a shared volume. Applications that write to themselves, for example to set user preferences, will

fail under these circumstances. Besides, there is a better way to save preferences, described in another one-liner.

Use PBGetVInfo to convert a VRefNum to a volume name.
References: IM IV-129

Delete uses the Poor Man's Search Path, so don't delete blindly.
References: IM IV-113, IM IV-147

As the high level FSDelete is an old MFS call, it will use the Poor Man's Search Path. This means that, if it does not find a file of the right name in the current directory to delete, it will try to delete such a file in the System Folder. This can be disastrous. Before you call Delete, make absolutely sure that it will delete the correct file by checking to see if such a file exists where you expect it.

File Manager routines with dirID=0 use the Poor Man's Search Path.

Actually, every File Manager routine with dirID=0 or no dirID specified uses the Poor Man's Search Path. Most of the time this just makes things more convenient, as it allows the System File to be searched by default after the current directory. Beware of what is happening, though.

Truncate and reallocate files before overwriting to reduce fragmentation.
References: IM IV-111, IM IV-112

As you overwrite a file again and again with sometimes less and sometimes more data, the file can become fragmented. Lots of fragmented files will worsen the performance of the entire file system. To help avoid this fragmentation, every time you need to overwrite a file from the beginning to the end, first truncate it to zero bytes using the SetEOF function and then allocate the number of bytes you expect to write. The FileManager will try to allocate a nice contiguous chunk of blocks.

Check/change the creator and type of Save As... files before overwriting.
References: IM IV-113

Just checking for the existence of a file with the same name is not enough. You must make sure that the file type and creator are correct as well. Otherwise, the file will not appear correct to the Finder, and you may not be able to read it in the future. There are a variety of strategies to do this. Some programs simply don't let you Save As over a file of a different type or creator. Some quietly change the type and creator. Some put up an alert. You decide.

If you rewrite files by deleting and creating, copy all Finder information.
References: IM IV-113

Deleting and recreating is not the best way to rewrite a file, but if you must do it, be sure to copy all the Finder information. This means the entire contents of the FInfo record. One more piece of information that needs to be copied is the GetInfo comment. I don't think there is a standard way of doing this, which is one of the reasons deleting and recreating is a bad thing to do.

Directory ID's are longs, not shorts. Shorts work ALMOST all the time.
References: IM IV-92

If a file version number appears in a file manager call, always set it to 0.
References: IM IV-90

The version number of a file was an old mechanism to distinguish between two files with the same name, for example to use one as a backup for the other. It was a good idea, but it was never completely implemented. The problem is that different bits of software do different things with the version number. The Finder ignores the version number, but the standard file routines only show files with version 0. This is the most common cause of bugs involving a file that you can see in the Finder but not within an application. To prevent this from happening, if a version number appears in a call, always explicitly set it to 0 before you make that call.

To convert a pathRefNum to a name or file number, use PBGetFCBInfo.
References: IM IV-179

Prevent the creation of files with names that begin with a period.
References: IM II-175, IM II-245

The File Manager interprets any file name beginning with a period as the name of a device driver. Unfortunately, it is also possible to create real files with names that begin with a period, causing no end of confusion. To protect your users from this, add a check to your Save As routines to reject file names that begin with a period.

Write/update the Finder info before writing data or resource forks.
References: IM IV-113

Lock handles before passing their dereferenced pointers to any routine.
References: IM XREF

It is well known that some Toolbox routines are known to change memory. The Inside Macintosh X-Ref has a list of such routines, which it claims is complete. Hah! The list is really growing all the time, and you cannot count on any routine's being safe any more. Even if you could, you run the risk that your language accesses that routine through glue which can be loaded the first time you call the routine, thus changing memory. The best way to be safe from this is to lock all handles before dereferencing their pointers and passing them to any routine, however benign the routine may seem.

Lock handles before setting referenced data to expressions containing functions

This is a nasty one. Let us say you have a C statement like `(*aHandle)[5] = foo(3)`; in other words, aHandle is a handle to an array, and you want to set element number 5 of that array to the return value of foo. This is unsafe! The reason is that C will calculate the pointer to the fifth element of aHandle before it calls foo. If foo compacts memory, this pointer will no longer be valid. This problem is not limited to C; it can occur with any language that allows similar constructs. Don't count on a certain order of evaluation. Lock the handle, or use an intermediate variable.

Put an odd long at location zero on a 68000 to help find NIL handle references.

Whenever you use a NIL handle or pointer, it will look at whatever is stored in location 0. The long word at location 0 could conceivably point anywhere, causing all sorts of indirect problems to result from nil handle references. If you suspect your application has NIL handle references, you can track them down by using a Macintosh that uses the old 68000 processor. If you put an odd value at location 0, whenever your application tries to use a NIL handle, you will get an exception immediately, and you will break into the debugger.

Call MoveHHi before locking a handle to avoid memory fragmentation.
References: IM II-44

If you lock a handle, it may be in the middle of the heap. If you lock several handles, they may fragment memory and affect subsequent memory allocations. You can avoid this by first calling MoveHHi on the handle before locking it. MoveHHi will take some time, but it will try to move the handle as high in memory as possible before you lock it.

Use HGetState/HLock/HSetState to lock a handle then put it back as it was.
References: IM IV-79

Most of the time, the best way of using handles is always to keep them unlocked and only lock them when it is absolutely necessary for a short time. However, sometimes you have a handle that you don't know is locked and you need to lock it and then set it back to the previous state. To do this, use HGetState to get the state of the handle before locking it. Then, instead of using HUnlock, use HSetState to restore the handle to its previous state.

Always use unsigned characters within text and Pascal-format strings.

The first character of a Pascal format string is its length in bytes, from 0 to 255. This length only makes sense if you are using unsigned characters. If you are using signed characters, such as the default C char, numbers above 127 will be interpreted as negative numbers. This is very dangerous, especially when you use packages such as TextEdit, which take a length. Most languages will quietly promote a signed

character to a signed short or long and will happily pass this value to TextEdit, which will interpret it as a VERY LARGE length, whereupon TextEdit will crash and burn. To avoid this, always use unsigned characters.

Save application preferences in a folder named Preferences in the System Folder

As it is a bad idea to keep application preferences in the file itself, where do you keep them? After a discussion of this matter on Usenet, the consensus seemed to be that there should be a folder named Preferences in the System Folder and each application should have file with an application-specific name in that folder. To make it as general as possible, the algorithm should work like this: First look in the current directory for the preferences file. If it is found, use that. The Poor Man's Search Path will ensure that any file in the System Folder will be found as well. If no file is found, look for a Preferences folder in the System Folder and look for the file there. If none is found, search all folders in the System Folder for the file. (This allows the user to rename the Preferences folder.) If it is not found, create a Preferences folder and save a file with the default preferences in that folder.

Use SysEnvirons to find the System (Blessed) Folder.
References: IM V-5

Use GetAppParms to get the name of the application.
References: IM II-58

The high bit of SysParam . volClik enables the alarm clock.
References: IM II-370

Check the application name at $910 before exiting with ES within MacsBug.

Hitting the programmer's interrupt switch and doing an ES within MacsBug is a good way to stop a runaway application. Unfortunately, with MultiFinder, you never know just what is running when you hit that switch. To find out, look at location $910 to see the name of the current application. If it is not the one you want to stop, enter G and try again.

To exit to shell in the mini-debugger, enter SM 0 A9 F4 and then G 0.

The mini-debugger has very few commands. One of the most useful functions, exit to shell, is not provided. There are a lot of ways of doing an exit to shell, but most of them involve remembering magic numbers of addresses for different systems. The way presented here should work on any machine. A9F4 is the code for the ExitToShell trap. SM 0 A9 F4 puts that instruction in the RAM at location 0. G 0 jumps to that instruction and executes it.

In Pascal, don't nest procedures to be passed by procedure pointer.

In Pascal, "with theHandle^^" is unsafe if memory compaction can occur.

Because a with statement dereferences a handle at the beginning of the block of code it controls, if the code causes a memory compaction, this handle may no longer be valid. Lock the handle first.

Avoid writing tail patches for traps.
References: TN 212

Some Apple patches to traps check the stack to see who called them and have some special cases. I know that this is not very sociable of them, but the upshot is that your application will get blamed if you use a tail patch and this causes one of Apple's bug fixes to fail.

There is no official way to tell if MultiFinder is running or not.

I wish there were a way to tell this, because it affects the behavior of Launch, but there isn't. There have been a variety of unofficial ways of telling, but most of them have failed as the system was changed. The only absolutely safe way to tell seems to be to assume that you are not running under MultiFinder until you receive a MultiFinder event.

Scheme to Manage a "Windows" menu

by Ben Cranston

Back in August of 1989 there was a discussion here on methods for implementing a "Windows" menu, one with an entry for each window currently displayed by the application. The act of selection would bring that window to the fore, etc.

David Phillip Oster asked: "what happens if you have multiple files open, all with the same name?" and then suggested "perhaps the simplest solution is to just use the item position in the windows menu, rather than the title, to determine which window the user wishes to select -- a problem with this is that you can get identical menu items, but at least the program can distinguish among them, if not the user."

This posting describes a simple scheme to implement this paradigm. Please feel free to use your "junk" key if you are not interested.

The scheme comprises two parts:

1. A linked list of the windows, sorted by the window names, is kept from a global cell through a window pointer variable in each window data area. Note: this list is in addition to the Window Manager's list.

2. A small integer in the window data area holds the index in the menu of that window's item. I called it MRefNum ("Menu Reference Number), for lack of a better name.

The management algorithms are:

When a new window is created, the proper place in the linked list must be found. We scan the linked list until we find either the end of the list or a window whose name is lexically greater than that of our new window. We are then interested in the item BEFORE that position which is either the root or a window whose name is lexically less than the new name.

The MRefNum of this preceding entry determines the position in the menu that the new window's name is inserted. It is also incremented and becomes the MRefNum of the new window. The new window is inserted into the chain at this point, and all succeeding windows in the chain have their MRefNum's incremented, corresponding to the fact that their menu entries were pushed down by the insert of the new window name.

If the previous item was the root, the new MRefNum becomes 1, and the same things are done.

When a window is destroyed, the menu item corresponding to its MRefNum is deleted, the window is removed from the chain, and all succeeding windows in the chain have their MRefNum's decremented.

I did have an algorithm for changing the name, which broke down into two possibilities corresponding to moving a name UP in the chain (incrementing the MRefNums between the new and old position) or moving a name DOWN in the chain (decrementing the MRefNums between the old and new position) but in the actual program I deleted and re-inserted the window :-)

Getting from a window to it's menu item is easy, as the MRefNum designates the appropriate menu index. Getting from a menu item to the window is just searching for the appropriate MRefNum value.

Here's the code for creating a window:

```
/* Insert a window into the proper point in the window chain.
 * Also makes appropriate entry in the window menu.
 */
```

```
InsertWind(RWPtr newwind)
{
    RWPtr   lastw = nil;
    RWPtr   nextw = AFWind;  /* the root */
    short   refn;

    while ( (nil!=nextw) && (0<CompHand(newwind->wname,nextw->wname)) ) {
    lastw = nextw;
    nextw = nextw->next;
    }

    refn = (nil!=lastw)?lastw->mrefn:0;
    newwind->mrefn = refn+1;
    newwind->next = nextw;

    if (lastw!=nil)
    lastw->next = newwind;
    else
    AFWind = newwind;

    InsMenuItem(WMenu,"\pa",WMBASE+refn);
    SetItem(WMenu,WMBASE+newwind->mrefn,*newwind->wname);

    while (nil != nextw) {
    nextw->mrefn++;
    nextw = nextw->next;
    }

}
```

Here's the code for destroying the window:

```
/* This routine removes a window from the window chain.
 * It also removes the appropriate window menu entry.
 */

RemoveWind(RWPtr mortwind)
{
    RWPtr   lastw = nil;
    RWPtr   nextw = AFWind;  /* the root */

    while ( (nil!=nextw) && (nextw!=mortwind) ) {
    lastw = nextw;
    nextw = nextw->next;
    }

    if (nil!=nextw) {
    if (nil!=lastw)
        lastw->next = nextw->next;
    else
        AFWind = nextw->next;
    DelMenuItem(WMenu,WMBASE+mortwind->mrefn);
    while (nil!=nextw) {
        nextw->mrefn--;
        nextw = nextw->next;
    }
    } else
    SysBeep(30);
}
```

Here's some code (case WIMENU) that goes from the menu item to the window.

```
/* Process selection from menu.
 * Apple menu:
 *     Note: MUST do DA stuff so MF can switch from Apple menu...
```

```
 *     Note: no "about" box is implemented.
 * FILE menu:
 *  ...
 * SHOW menu:
 *  Flip view bits or change displayed resource type.
 * WIND menu:
 *  Iconize/deiconize or bring window to front.
 */

DoMenu(int menuparam)
{
    char       daname[256];
    int        menuitem = LoWord(menuparam);
    WindowPtr  windp = FrontWindow();
    RWPtr      mywind = windp;
    short      wkind = WindowKind(windp);

    switch ( HiWord(menuparam) ) {
    case APMENU:
    if ( menuitem == 1 )
        SysBeep(30);
    else {
        GetItem(AMenu,menuitem,daname);
        OpenDeskAcc(daname);
        AdjustMenus();
    }
    break;
    case EDMENU:
    SystemEdit(menuitem-1);
    break;
    case FIMENU:
    switch (menuitem) {
    case FMOPEN:
        OpenWindow();
        break;
    case FMCLOS:
        if (wkind < 0)
       CloseDeskAcc(wkind);
        else
       KillWindow(windp);
        break;
    case FMPSET:
        PrintSetup(windp);
        break;
    case FMPRNT:
        PrintWindow(windp);
        break;
    case FMQUIT:
        ExitToShell();
    }
    break;
    case SHMENU:
    if (nil != windp) {
        if (menuitem >= SMBASE) {
       OpenType = menuitem-SMBASE;
       SetWindType(windp,OpenType);
        } else switch (menuitem) {
        case SMRNUM:
       mywind->wview.show.rnum = mywind->wview.show.rnum ^ 1;
       break;
        case SMRNAME:
       mywind->wview.show.rname = mywind->wview.show.rname ^ 1;
       break;
        case SMMASK:
       mywind->wview.show.mask = mywind->wview.show.mask ^ 1;
        }
```

```
        PostWindowSize(mywind);
        SetPort(windp);
        InvalRect(&windp->portRect);
        AdjustMenus();
    }
    break;
    case WIMENU:
    if (menuitem<WMBASE) {
        if (nil != windp) {
        switch (menuitem) {
        case WMICZE:
            if (mywind->wview.show.icize)
            DeIconizeWindow(mywind);
            else
            IconizeWindow(mywind);
        }
        SetPort(windp);
        InvalRect(&windp->portRect);
        AdjustMenus();
        }
    } else {
        for ( mywind=AFWind ; mywind!=nil ; mywind=mywind->next)
        if (WMBASE+mywind->mrefn == menuitem)
            SelectWindow( (WindowPtr) mywind);
    }

    }
    HiliteMenu(0);
}
```

Here's some code (last for loop in this proc) that gets from window to menu:

```
/* Set checkmarks of view and window menus according to mode of front window.
 * There are three cases:
 * If FrontWindow() is nil then there are no windows open (degenerate case).
 * If FrontWindow() is not nil but windowKind is negative then the window is
 *  a Desk Accessory opened by (single) Finder or a DA opened by MultiFinder
 *  within our application heap (option launch).  In this case we want to
 *  make the EDIT menu active but not any of our own menus, since the window
 *  will not have the data items our code assumes are there.
 * If windowKind is positive we assume it is one of our own normal windows,
 *  since we do not have any modeless dialogs.
 */

AdjustMenus()
{
    WindowPtr  windp = FrontWindow();
    RWPtr      mywind = windp;
    short      indx;

    if (nil == windp) {
    DisableItem(FMenu,FMCLOS);
    DisableItem(FMenu,FMPSET);
    DisableItem(FMenu,FMPRNT);
    DisableItem(EMenu,0);
    DisableItem(SMenu,0);
    CheckItem(SMenu,SMRNUM,false);
    CheckItem(SMenu,SMRNAME,false);
    CheckItem(SMenu,SMMASK,false);
    for (indx=0 ; indx<NRType ; indx++)
        CheckItem(SMenu,SMBASE+indx,false);
    DisableItem(WMenu,0);
    CheckItem(WMenu,WMICZE,false);
    } else if ( WindowKind(windp) < 0 ) {
    EnableItem(FMenu,FMCLOS);
    DisableItem(FMenu,FMPSET);
```

```
    DisableItem(FMenu,FMPRNT);
    EnableItem(EMenu,0);
    DisableItem(SMenu,0);
    CheckItem(SMenu,SMRNUM,false);
    CheckItem(SMenu,SMRNAME,false);
    CheckItem(SMenu,SMMASK,false);
    for (indx=0 ; indx<NRType ; indx++)
        CheckItem(SMenu,SMBASE+indx,false);
    DisableItem(WMenu,0);
    CheckItem(WMenu,WMICZE,false);
    } else {
    EnableItem(FMenu,FMCLOS);
    EnableItem(FMenu,FMPSET);
    EnableItem(FMenu,FMPRNT);
    DisableItem(EMenu,0);
    EnableItem(SMenu,0);
    CheckItem(SMenu,SMRNUM,mywind->wview.show.rnum);
    CheckItem(SMenu,SMRNAME,mywind->wview.show.rname);
    CheckItem(SMenu,SMMASK,mywind->wview.show.mask);
    for (indx=0 ; indx<NRType ; indx++)
        CheckItem(SMenu,SMBASE+indx,indx==mywind->wdata.dtype);
    EnableItem(WMenu,0);
    CheckItem(WMenu,WMICZE,mywind->wview.show.icize);
    }

    for ( mywind=AFWind ; mywind!=nil ; mywind=mywind->next)
    CheckItem(WMenu,WMBASE+mywind->mrefn,( (RWPtr) windp == mywind ));

    DrawMenuBar();
}
```

These are subroutines used by the above:

```
/* Compare string handles
 */

CompHand(Handle s1,Handle s2)
{
    int ans;

    HLock(s1);
    HLock(s2);
    ans = IUCompString(*s1,*s2);
    HUnlock(s1);
    HUnlock(s2);
    return(ans);
}

/* Get the windowKind field from the window record.  This is used to
 * decide if a window belongs to the program, or if it is a desk
 * accessory called either from the old Finder environment or from
 * the MultiFinder environment with the option key down.
 */

WindowKind(WindowPtr windp)
{
    int wkind = 0;

    if (nil != windp)
    wkind = ((WindowPeek) windp)->windowKind;
    return(wkind);
}
```

Actually, it occurs to me that there is no need to explicitly store the MRefNum at all, as it should be identical to the position of that window in the list (that is, reading the list should return 1, 2, 3, etc). Extension to remove the MRefNum cell is left as an exercise to the reader...

How to write an INIT in Pascal

by Matthew Xavier Mora

This started out to be an "is it possible to write an INIT in PASCAL?" project to see if I could do It. The answer is yes and no. It can be done but you will need either some inline assembly or some assembly glue code. The INIT I wrote is a jGNEfilter that checks to see if the user hit one of the extended keyboard function keys. The unit "newJGNEFilter" is not real working code. It is only a framework of what you need to do. I myself have not finished the code yet so I am not sure if it will be the way I go. When my INIT is finished I will update this article to include full working code.

```
{This unit is the code that will install a jGNEFilter patch.}
{Written by Matthew Xavier Mora}

unit install;
interface
    procedure main;

implementation
    procedure ShowINIT (iconID, moveX: Integer);
    EXTERNAL;
    procedure main;
        var
            newjgne: Handle;
            addr: longint;
            JGNE: ptr;
            jgneptr: ^ptr;
    begin
        SetZone(SystemZone);
        newjgne := Get1Resource('CODE', 128);
        if newjgne <> nil then
            begin
                DetachResource(newjgne);
                HLock(newjgne);
                JGNE := pointer($29A);
                BlockMove(JGNE,ptr(ord(newjgne^)+10),4);{move jgne address into header of
newcode}
                jgneptr := pointer($29A);
                jgneptr^ := pointer(newjgne^);
                ShowINIT(128, -1);
            end;
    end;
end.

unit newJGNEFilter;

interface

    procedure main;

implementation
    function GetA1 (dummy: longint): longint;
    external;
    function GetD0 (dummy: longint): integer;
    external;
    procedure Setresult (dummy: integer);
    external;
    procedure DoJsr (addr: ProcPtr);
    inline
        $205F, $4E90;

    procedure DoJmp (addr: ProcPtr);
    inline
```

```
    $4CDF, $1CE0, $4E5E, $205F, $4ED0;
           {MOVEM.L    (A7)+,D5-D7/A2-A4}
           {UNLK       A6}
           {MOVEA.L    (A7)+,A0}
           {JMP        (A0)}


procedure main;

    var

        addr, oldjgne: procptr;
        dummy: integer;
        Eventmessage: longint;
        event: eventrecord;
        hasevent: boolean;
        Evntptr: ^eventrecord;
begin
    hasevent := Boolean(GetD0(0));         {D0 contains flag of gne}
    if hasevent then
        begin
            Evntptr := pointer(geta1(0));   {A0 contains a pointer to the eventrecord}
            Eventmessage := Evntptr^.message;
            {do whatever you wish }
            {if you hande the event your self then set D0 to false}
            { SetD0(false);}
        end;
    oldjgne := procptr(ord(@main) - 6);     {get address that was stored into header}
    addr := procptr(oldjgne);
    if addr <> nil then
        DoJmp(addr);                        {jmp to address stored in header}
end;
```

How do you play Asynchronous Sound?

by Larry Rosenstein

[Asynchronous Sound Code]

Enclosed is the source for an MPW Pascal unit that shows how to play asynchronous sounds with the Sound Manager. I have tried this unit only on System 6.0.2; supposedly there are bugs in earlier versions of the Sound Manager. This unit also doesn't check for the existence of the Sound Manager, I assume that you do this at a higher level.

I used this in a simple MacApp program that will open any file and allow you to play any snd resource in the file ansynchronously. (I started this with the idea of allowing copy & paste, but haven't gotten that far yet. If there is interest, I can post that program.)

Larry Rosenstein, Object Specialist

```
---
(*
A simple unit that demonstrates how to produce asynchronous sound with
the Macintosh Sound Manager.  Although I am pretty confident about this code,
I don't guarantee that this code demonstrates the correct way to do things.  It
does seem to work reliably.

This unit doesn't solve any of the tricky issues about using the Sound Manager.
Primarily, sound channels should be disposed of as soon as they are no longer
needed.  This code does just that, but it doesn't prevent your program or a
background program from trying to make sound.

Changes:

2/16/89  Lock the sound resource; state restored in call back
gSoundPlaying is the actual handle.

Larry Rosenstein
Apple Computer, Inc.
lsr@Apple.COM

Copyright 1988-1989 Apple Computer Inc.  All Rights Reserved.
*)

UNIT UAsynchSnd;

INTERFACE

USES
MemTypes, Quickdraw, OSIntf, ToolIntf;

{ call this before any other routine }
PROCEDURE InitUAsynchSnd;

{ returns TRUE if an asynchronous sound is playing }
FUNCTION  IsSoundPlaying: BOOLEAN;

{ equivalent to SndPlay, but does it asynchronously; if you call this
while another sound is playing, the first one will be stopped }
FUNCTION  ASynchSndPlay(sndHandle: Handle): OSErr;

{ stop the sound from playing; may be called even if no sound is
currently playing }
PROCEDURE StopAsynchSound;

{ should be called when your program exits }
```

```
PROCEDURE TerminateUAsynchSnd;

IMPLEMENTATION

VAR
gSoundPlaying:          Handle;
gSoundState:            SignedByte;
gSndChannel:            SndChannelPtr;

PROCEDURE ChanCallBack(chan: SndChannelPtr; cmd: SndCommand); FORWARD;
FUNCTION  GetA5: LONGINT; INLINE $2E8D; {MOVE.L A5,(A7)}
FUNCTION  LoadA5(newA5: LONGINT): LONGINT;
                                        INLINE $2F4D,$0004,$2A5F;

(********************)

PROCEDURE InitUAsynchSnd;
BEGIN
gSndChannel := NIL;
gSoundPlaying := NIL;
END;

FUNCTION  ASynchSndPlay(sndHandle: Handle): OSErr;
VAR     err:                    OSErr;
aCommand:               SndCommand;

BEGIN
StopAsynchSound;        { kill the current sound & channel }

err := noErr; { default value }

{ gSndChannel should be NIL now }
err := SndNewChannel(gSndChannel, 0, 0, @ChanCallBack);
{ We don't specify a synthesizer, since we are assuming that
the snd resource specifies one.  For example, the
standard Clink-Klank snd resource doesn't use the
sampled synthesizer. }

gSoundState := HGetState(sndHandle);
MoveHHi(sndHandle);
HLock(sndHandle);
gSoundPlaying := sndHandle;

IF err = noErr THEN
err := SndPlay(gSndChannel, sndHandle, TRUE);

WITH aCommand DO BEGIN
cmd := callBackCmd;
param1 := 0;
param2 := GetA5;
END;
IF err = noErr THEN
err := SndDoCommand(gSndChannel, aCommand, FALSE);

IF err <> noErr THEN
StopAsynchSound;                { flush channel; unlock sound }

AsynchSndPlay := err;
END;

PROCEDURE ChanCallBack(chan: SndChannelPtr; cmd: SndCommand);
VAR     oldA5:  LONGINT;
BEGIN
oldA5 := LoadA5(cmd.param2);    { get the application's A5 and set it }

HSetState(gSoundPlaying, gSoundState);
```

```
gSoundPlaying := NIL;

oldA5 := LoadA5(oldA5);                 { restore old A5 }
END;

FUNCTION  IsSoundPlaying: BOOLEAN;
BEGIN
IsSoundPlaying := gSoundPlaying <> NIL;
END;

PROCEDURE StopAsynchSound;
BEGIN
IF gSndChannel <> NIL THEN BEGIN
IF SndDisposeChannel(gSndChannel, TRUE) = noErr THEN { nothing };
gSndChannel := NIL;
END;
IF gSoundPlaying <> NIL THEN
BEGIN
HSetState(gSoundPlaying, gSoundState);
gSoundPlaying := NIL;
END;
END;

PROCEDURE TerminateUAsynchSnd;
BEGIN
StopAsynchSound;
END;

END.
```

Default 2.1 CDEF

By Lloyd Lim

The Default CDEF is a simple aid for Macintosh programmers that draws default button outlines for any size buttons, in the proper color, in your application and in ResEdit dialog and alert templates. Push buttons, check boxes, and radio buttons can also be drawn using the window's font.

Instructions

Default is a control definition function that you can copy and paste into your application. The Default CDEF enhances the System file's standard control definition 0. If the last character of a button's title is an '@' (an at or an each symbol), the button is drawn with an outline indicating that it is the default button. The trailing '@' is not drawn with the title.

If the button is inactive, the outline is grayed out. When Color QuickDraw is available, outlines are drawn in the same color as the button's frame. If the Default CDEF is in the same file as your application's DITL resources, ResEdit will display default button outlines drawn by Default. You do not need to write any code to use Default.

A dialog's default button can be changed at any time in your application simply by changing the appropriate button titles. However, you, the programmer, are responsible for making sure there is only one default button. If you are not using a filterProc, Default makes sure that pressing the Return key or Enter key is the same as clicking in the default button. The default button is also highlighted when the Return key or Enter key is pressed. If you are using your own filterProc, you must perform these tasks.

If the last character of a control's title is an '*f*', the control is drawn using the window's current font. Push buttons, check boxes, and radio buttons will look the same as if you used a CNTL resource with the useWFont variation code. This feature is especially useful for CDEVs. If you need a default button that uses the window's font, end the button's title with '*f*@'.

Compatibility

Default should work on any model of Macintosh with any System version. Default is 32-bit clean. Except for drawing the outline, Default lets the System file's CDEF 0 do practically everything. Thus, Default's buttons do anything that normal buttons do.

Be warned that Default uses the contrlData field of a ControlRecord (which is okay since it is reserved for CDEFs). If your application does something strange with this field, Default will not work.

When you drag or resize a default button in ResEdit, you may notice some slight flickering. This occurs because ResEdit does not know about the outline so Default forces ResEdit to refresh portions of the window. Default only behaves this way in ResEdit. This does not occur in normal applications. Default now comes in versions with and without the ResEdit updating code. This may be helpful if you are concerned about your application's size.

Bugs

The outline will not update if the outline needs updating and the button does not, and UpdtControl or UpdtDialog is being used to update the controls. This problem does not occur with normal modal dialogs and occurs rarely with modeless dialogs. Unfortunately, there doesn't seem to be a clean solution this problem. If you have this problem, you can call Draw1Control for the default button on every update event or simply use DrawControls or DrawDialog instead.

If you have a default button and you aren't using a filterProc, pressing the Return key or Enter key will click in the default button even if the button is hidden. This is a bug in the Dialog Manager. If you have this problem, you must change the default button to a normal button before you call HideDItem or HideControl.

Please report any other bugs to any of the addresses listed below. Thanks to those programmers who did unusual things with their buttons, reported bugs, and helped make Default more robust.

History

1.0 — first release version
1.1 — fixed bug with HideControl and default buttons
1.2 — updated to support new 32-bit clean CDEF messages
1.3 — fixed bug with DragControl
1.4 — fixed bug converting between default buttons and normal buttons;
improved outlines for unusually sized default buttons
1.5 — forced ResEdit to refresh default buttons correctly
2.0 — added automatic Return and Enter support for default buttons;
added window font feature for push buttons, check boxes, and radio buttons;
restructured code to reduce size of CDEF
2.1 — updated to follow Apple's new way of drawing default outlines

Distribution

Lim Unlimited
Postal: 330 W. Iris, Stockton, CA 95210, U.S.A.
Internet: lim@iris.ucdavis.edu
America Online: LimUnltd
CompuServe: 72647,660

```
/*
    Default is a CDEF written using THINK C. This CDEF replaces the standard CDEF 0 and
    simply draws a default button outline and calls the standard CDEF 0 to handle
    everything else.

    © Lim Unlimited, 9 Jul 1990 - All Rights Reserved
*/



/*
    header files
*/

#include <Color.h>
#include <ColorToolbox.h>
#include <FontMgr.h>



/*
    constants and macros
*/

#define RESEDIT   1      /* whether to compile ResEdit updating code */
```

```
#define NIL       ((void *) 0)

#define INACTIVE  255

#define STANDARD_CDEF_ID 0

enum {                    /* new 32-bit clean messages */
    calcCntlRgn = 10,
    calcThumbRgn
};

#define LO_3_BYTES    0x00FFFFFF

#define SYS_ENVIRONS_VERSION        1

#define DEFAULT_FLAG      '@'
#define USEWFONT_FLAG     'ƒ'

#define OUTLINE_THICKNESS    3
#define OUTLINE_INSET        (OUTLINE_THICKNESS + 1)
#define OUTLINE_OUTSET       (-OUTLINE_INSET)
#define CURVE_ADJUSTMENT 2

#define RECT_IN_RECT(r1, r2)
                  \
    ((r1)->top >= (r2)->top && (r1)->left >= (r2)->left &&                      \
     (r1)->bottom <= (r2)->bottom && (r1)->right <= (r2)->right)



/*
    typedefs
*/

typedef struct {
    ControlHandle  itmHndl;
    Rect               itmRect;
    Byte               itmType;
    Byte               itmData[1];
} Item, *ItemPtr;

typedef struct {
    short      dlgMaxIndex;
    Item       item[1];
} ItemList, *ItemListPtr, **ItemListHdl;

typedef struct {
    Handle     standardCDEF;
    unsigned       has128KROMS:1;
    unsigned       hasColorQD:1;
    unsigned       dialogButton:1;
    unsigned       dialogDefault:1;
#if RESEDIT
    unsigned       inResEdit:1;
#endif
} DefaultData, *DefaultDataPtr, **DefaultDataHdl;



/*
    routines
*/

PASCAL long           main                                      (short, ControlHandle,
short, long);
```

```
/*
    static routines
*/

static long           CallStandardCDEF             (short, ControlHandle, short, long);
static Boolean        RealHandle                          (long, Boolean);
static short      FindItemNum                             (DialogPeek, ControlHandle);



/*
    static variables
*/

static unsigned char     Copyright[] = "Default 2.1, ©1990 Lim Unlimited";



/*
    main draws the default button outline and calls the standard button CDEF.
*/

PASCAL long main(varCode, theControl, message, param)

register short              varCode;
register ControlHandle   theControl;
register short              message;
register long            param;

{
    register DefaultDataHdl         defaultDataHdl;
    unsigned char                   *title;
    DialogPeek                             theDialog;
    Boolean                                isDefault, useWindowFont, dialogDefault;
    short                                  oldRefNum;
    Handle                          standardCDEF;
    SysEnvRec                       sysEnvirons;
    Rect                                   button;
    AuxCtlHndl                             auxCtlHdl;
    RGBColor                               oldColor, frameColor;
    PenState                               penState;
    Pattern                                gray;
    RgnHandle                       outlineRgn;
    register short                         curve;
    register long                   result;
#if RESEDIT
    RgnHandle                       visRgn, oldVisRgn;
    Rect                                   visButton, visBounds;
#endif

    title = (*theControl)->contrlTitle;
    if (!(varCode & ~useWFont) && title[title[0]] == (unsigned char) DEFAULT_FLAG) {
        isDefault = TRUE;
        --title[0];
    } else {
        isDefault = FALSE;
    }
    if (useWindowFont = (title[title[0]] == (unsigned char) USEWFONT_FLAG)) {
        varCode |= useWFont;
        --title[0];
    }
    theDialog = (DialogPeek) (*theControl)->
;
```

```
    if (message == initCntl) {
        defaultDataHdl = (DefaultDataHdl) NewHandle(sizeof(DefaultData));
        (*theControl)->contrlData = (Handle) defaultDataHdl;

        oldRefNum = CurResFile();
        UseResFile(0);
        standardCDEF = GetResource('CDEF', STANDARD_CDEF_ID);
        (*defaultDataHdl)->standardCDEF = standardCDEF;
        UseResFile(oldRefNum);
        HNoPurge(standardCDEF);

        result = SysEnvirons(SYS_ENVIRONS_VERSION, &sysEnvirons);
        (*defaultDataHdl)->has128KROMS = (sysEnvirons.machineType >= envMachUnknown);
        (*defaultDataHdl)->hasColorQD = sysEnvirons.hasColorQD;
        if (!(varCode & ~useWFont) &&
             RealHandle((long) theDialog->items, !result) &&
             RealHandle((long) theDialog->textH, !result) &&
             theDialog->editField >= -1 &&
             theDialog->editField <= (*(ItemListHdl) theDialog->items)->dlgMaxIndex) {
            (*defaultDataHdl)->dialogButton = TRUE;
        } else {
            (*defaultDataHdl)->dialogButton = FALSE;
        }
        (*defaultDataHdl)->dialogDefault = FALSE;

#if RESEDIT
        /* ResEdit is being used if the control's window has no controls attached */
        if (!theDialog->window.controlList && theDialog == (DialogPeek) FrontWindow()) {
            (*defaultDataHdl)->inResEdit = TRUE;
        } else {
            (*defaultDataHdl)->inResEdit = FALSE;
        }
#endif

        result = CallStandardCDEF(varCode, theControl, message, param);

    } else {
        defaultDataHdl = (DefaultDataHdl) (*theControl)->contrlData;

        /* set default button item number for Dialog Manager */
        if ((*defaultDataHdl)->dialogButton) {
            dialogDefault = (isDefault && !(*theControl)->contrlHilite);
            if (dialogDefault && !(*defaultDataHdl)->dialogDefault) {
                theDialog->aDefItem = FindItemNum(theDialog, theControl);
                if (theDialog->aDefItem) {
                    (*defaultDataHdl)->dialogDefault = TRUE;
                }
            } else if (!dialogDefault && (*defaultDataHdl)->dialogDefault) {
                if (theDialog->aDefItem == FindItemNum(theDialog, theControl)) {
                    theDialog->aDefItem = 0;
                }
                (*defaultDataHdl)->dialogDefault = FALSE;
            }
        }

        if (message == drawCntl) {

            result = CallStandardCDEF(varCode, theControl, message, param);
            if (isDefault && (*theControl)->contrlVis) {
#if RESEDIT
                /* portions of the ResEdit window are invalidated under certain
                    circumstances so it will update correctly */
                if ((*defaultDataHdl)->inResEdit) {
                    visRgn = theDialog->window.port.visRgn;
                    visBounds = (*visRgn)->rgnBBox;
```

```
                button = (*theControl)->contrlRect;
                if (SectRect(&button, &theDialog->window.port.portRect, &visButton) &&
                     RECT_IN_RECT(&visButton, &visBounds)) {
                    InsetRect(&button, OUTLINE_OUTSET, OUTLINE_OUTSET);
                    if (SectRect(&button, &theDialog->window.port.portRect, &visButton) &&
                         !RECT_IN_RECT(&visButton, &visBounds)) {
                        outlineRgn = NewRgn();
                        oldVisRgn = NewRgn();
                        CopyRgn(visRgn, outlineRgn);
                        CopyRgn(visRgn, oldVisRgn);
                        RectRgn(visRgn, &theDialog->window.port.portRect);
                        InsetRgn(outlineRgn, OUTLINE_OUTSET, OUTLINE_OUTSET);
                        DiffRgn(outlineRgn, oldVisRgn, outlineRgn);
                        EraseRgn(outlineRgn);
                        CopyRgn(oldVisRgn, visRgn);
                        DisposeRgn(oldVisRgn);
                        InvalRgn(outlineRgn);
                        DisposeRgn(outlineRgn);
                    }
                }
            }
#endif

            if ((*defaultDataHdl)->hasColorQD) {
                (void) GetAuxCtl(theControl, &auxCtlHdl);
                GetForeColor(&oldColor);
                if (auxCtlHdl) {
                    frameColor = (*(*auxCtlHdl)->acCTable)->ctTable[cFrameColor].rgb;
                    RGBForeColor(&frameColor);
                }
            }

            GetPenState(&penState);
            PenNormal();
            PenSize(OUTLINE_THICKNESS, OUTLINE_THICKNESS);
            if ((*theControl)->contrlHilite == INACTIVE) {
                *(unsigned long *) &gray[0] = *(unsigned long *) &gray[4] = 0xAA55AA55;
                PenPat(&gray);
            }
            button = (*theControl)->contrlRect;
            InsetRect(&button, OUTLINE_OUTSET, OUTLINE_OUTSET);
            curve = ((button.bottom - button.top) >> 1) + CURVE_ADJUSTMENT;
            FrameRoundRect(&button, curve, curve);
            SetPenState(&penState);

            if ((*defaultDataHdl)->hasColorQD) {
                RGBForeColor(&oldColor);
            }
        }

    } else if (message == calcCRgns ||
                message == calcCntlRgn ||
                message == calcThumbRgn) {
        result = CallStandardCDEF(varCode, theControl, message, param);
        if (isDefault && ((message == calcCRgns && param > 0) ||
                            message == calcCntlRgn)) {
            OpenRgn();
            button = (*theControl)->contrlRect;
            InsetRect(&button, OUTLINE_OUTSET, OUTLINE_OUTSET);
            curve = ((button.bottom - button.top) >> 1) + CURVE_ADJUSTMENT;
            FrameRoundRect(&button, curve, curve);
            CloseRgn((RgnHandle) param);
        }

    } else if (message == dispCntl) {
        result = CallStandardCDEF(varCode, theControl, message, param);
```

```
            /* MultiFinder loads and shares system resources in the system heap; make
               standard button CDEF purgeable only if it is in the application heap */
            standardCDEF = (*defaultDataHdl)->standardCDEF;
            if (HandleZone(standardCDEF) == ApplicZone()) {
               HPurge(standardCDEF);
            }
            DisposHandle(defaultDataHdl);

#if 0
        /* messages not used by Default but documented for completeness */
        } else if (message == testCntl ||
                    message == posCntl ||
                    message == thumbCntl ||
                    message == dragCntl ||
                    message == autoTrack) {
            result = CallStandardCDEF(varCode, theControl, message, param);
#endif

        /* pass along messages which are not used by Default or are currently undefined */
        } else {
            result = CallStandardCDEF(varCode, theControl, message, param);
        }
    }

    if (useWindowFont) {
        ++(*theControl)->contrlTitle[0];
    }
    if (isDefault) {
        ++(*theControl)->contrlTitle[0];
    }
    return(result);
}



/*
    CallStandardCDEF calls the standard button CDEF.
*/

static long CallStandardCDEF(varCode, theControl, message, param)

short             varCode;
ControlHandle  theControl;
short             message;
long              param;

{
    register DefaultDataHdl         defaultDataHdl;
    register Handle                 standardCDEF;
    register SignedByte             flags;
    register long                   result;

    defaultDataHdl = (DefaultDataHdl) (*theControl)->contrlData;

    /* load standard button CDEF if it was purged */
    standardCDEF = (*defaultDataHdl)->standardCDEF;
    if (!*standardCDEF) {
        LoadResource(standardCDEF);
        HNoPurge(standardCDEF);
    }

    /* save and restore handle state just in case control is reentrant */
    if ((*defaultDataHdl)->has128KROMS) {
        flags = HGetState(standardCDEF);
    } else {
```

```
        flags = (long) *standardCDEF >> 24;
    }
    HLock(standardCDEF);
    result = CallPascalL(varCode, theControl, message, param, *standardCDEF);
    if ((*defaultDataHdl)->has128KROMS) {
        HSetState(standardCDEF, flags);
    } else {
        *standardCDEF = (Ptr) (((long) *standardCDEF & LO_3_BYTES) | (flags << 24));
    }
    return(result);
}

/*
    RealHandle returns whether the given address is a handle in the system heap or
    application heap.
*/

static Boolean RealHandle(addr, hasStripAddr)

register long  addr;
Boolean           hasStripAddr;

{
    register Boolean  real;
    register THz      sysZone, applZone, heapZone;
    real = FALSE;
    addr = (hasStripAddr) ? StripAddress(addr) : addr & LO_3_BYTES;
    if (addr && !(addr & 1)) {
        sysZone = SystemZone();
        applZone = ApplicZone();
        if (((addr >= (long) &sysZone->heapData && addr < (long) sysZone->bkLim) ||
              (addr >= (long) &applZone->heapData && addr < (long) applZone->bkLim)) &&
             *(long *) addr && !(*(long *) addr & 1)) {
            heapZone = HandleZone(addr);
            if (!MemError() && (heapZone == sysZone || heapZone == applZone)) {
                real = TRUE;
            }
        }
    }
    return(real);
}




/*
    FindItemNum returns the item number of the given control in the given dialog.
*/

static short FindItemNum(theDialog, theControl)

DialogPeek        theDialog;
ControlHandle  theControl;

{
    register short        numItems, itemNum;
    register Ptr      itemPtr;

    numItems = (*(ItemListHdl) theDialog->items)->dlgMaxIndex + 1;
    itemPtr = (Ptr) (*(ItemListHdl) theDialog->items)->item;
    for (itemNum = OK; itemNum <= numItems; ++itemNum) {
        if (((ItemPtr) itemPtr)->itmHndl == theControl) return(itemNum);
        itemPtr += sizeof(Item) + (((long) ((ItemPtr) itemPtr)->itmData[0] + 1) & ~1);
    }
    return(0);
```

```
}
```

ToolBox Gotchas

by John Norstad

While developing Disinfectant I ran into a number of "gotchas" that caused me great grief. I thought it would be nice to tell the rest of you about these problems, in the hope that you'll be able to avoid them in your own programs. I've told DTS about most of this stuff.

Gotcha #1. Watch out for PBGetCatInfo calls with TOPS.

The file manager routine PBGetCatInfo uses a parameter block of type CInfoPBRec. Make certain that you pass a pointer to the full parameter block when using MPW C, even if you know in advance that the object is a directory. Don't just allocate and pass a pointer to the DirInfo variant. The DirInfo variant is four bytes shorter than the full union type, and with TOPS the PBGetCatInfo call sets those four bytes at the end. If your parameter block is not big enough you'll trash the stack.

Gotcha #2. Make certain you're in the proper heap zone before calling ReleaseResource.

At the bottom of IM II-26, in the Memory Mangler chapter, is the warning "Be sure, when calling routines that access blocks, that the zone in which the block is located is the current zone." Heed this warning, especially when releasing resources. Bob Hablutzel and I discovered (after hours in Macsbug) that on the 128K ROMs, if you try to release an empty (unloaded) resource in the system heap, and if you neglect to set the current zone to the system zone, then the system will trash the free master pointer list. This is not good, and will almost undoubtedly lead to subsequent bizarre behavior.

Here's the code I use to release a resource:

```
curZone = GetZone();
SetZone(HandleZone(theRez));
ReleaseResource(theRez);
SetZone(curZone);
```

Gotcha #3. Don't believe Inside Macintosh. (HandleZone)

On page IM II-34 we read the following warning in the description of the HandleZone routine: "If handle h is empty (points to a NIL master pointer), HandleZone returns a pointer to the current heap zone." This is false - HandleZone properly returns a pointer to the heap zone that contains the master pointer. See Gotcha #2 above.

Gotcha #4. Don't expect OpenResFile to do sanity checking.

Neither OpenResFile nor OpenRFPerm does any sanity checking of any sort when opening a resource file. If the file is damaged or contains trash it is very possible for the Resource Mangler to bomb or hang inside the OpenResFile or OpenRFPerm call. Often what happens is that it makes a Memory Mangler request for some ridiculously huge block of memory. If you have a GrowZone proc this can cause problems.

To prevent this problem you must write a sanity checker of your own that opens the resource file as a binary file and checks at least the most important structural characteristics of the file. If your sanity check fails you must avoid calling OpenResFile or OpenRFPerm on the file. In Disinfectant I check that the resource map and resource data are within the logical EOF of the file and don't overlap, I check that the resource type list immediately follows the resource map, and I check that the resource name list starts within the logical eof.

DTS tells me that the only way to be completely safe is to do a complete sanity check of the entire resource fork - e.g., rewrite the RezDet MPW tool.

Damaged and trashed resource forks are much more common than you might think.

Gotcha #5. Don't believe Inside Macintosh. (OpenResFile)

In the description of the OpenResFile routine, IM I-115 states "If the resource file is already open, it doesn't make it the current resource file; it simply returns the reference number." This is false. If the resource file is already open, OpenResFile in fact DOES make it the current resource file. OpenRFPerm also has the same behavior, in those cases where OpenRFPerm returns the reference number of the previously opened copy of the file, rather than opening a new access path (see IM IV-17 and TN 185).

Gotcha #6. Watch out for Standard File if you unmount volumes.

The standard file package keeps track of the last volume it used in the low core global SFSaveDisk, which contains the negative of the vol ref num of the last volume used. If your program unmounts this volume and then later calls the standard file package again, it will post an alert saying that "A system error has occurred. Please try again." A simple fix for this problem is to check the vRefNum stored in SFSaveDisk immediately before any calls to standard file. Call PBGetVInfo to see if the volume still exists. If it doesn't, make an indexed call to PBGetVInfo to get the vRefNum of the first volume in the VCB queue, and set SFSaveDisk to the negative of this vRefNum. Also set CurDirStore to fsRtDirID.

Gotcha #7. Don't believe Inside Macintosh.

IM I-116 states that "When calling the CurResFile and HomeResFile routines, described below, be aware that for the system resource file the actual reference number is returned." This is false. CurResFile does indeed return the actual reference number of the system file (2), but HomeResFile in fact returns 0 for system file resources.

Gotcha #8. Don't believe Inside Macintosh.

IM I-126 states "Like the attributes of individual resources, resource file attributes are specified by bits in the low-order byte of a word." This is false. In fact, the resource file attributes are stored in the high-order byte of the word.

Gotcha #9. Directory IDs are longs, not shorts, stupid.

Directory IDs, unlike volume reference numbers and working directory ids, are longs, not shorts. Watch out for this one. It's really easy to declare a dirID to be a short by mistake, and unless you're using Modula-2 you probably won't catch the bug even with extensive beta testing. Don't feel too stupid if you do this - I have it on good authority that ResEdit once had this bug!

Gotcha #10. Always set the ioNamePtr field in file manager param blocks.

See TN 179. Read it. Believe it. Always set ioNamePtr. Set it to nil if you don't care about the name. I made this mistake three times while developing Disinfectant, and all three times it took FOREVER to find the bug. The problem is those silly little arrows in the file manager chapter of IM IV. They all point to the left for ioNamePtr, which usually means that you don't have to set the field before calling the routine.

I hope my experiences help somebody.

John Norstad Academic Computing and Network Services Northwestern University

How do you Hide the menu bar?

by: Earle R. Horton

The code I posted yesterday would cause problems in the unlikelyevent that your program would crash while the Menu Bar was hidden. Specifically, it would replace GrayRgn with a handle to a Region in the application heap, and save the real GrayRgn Handle for restoration later when you restored the Menu Bar. If your program crashed, and the Menu Bar was hidden, then GrayRgn was left pointing to a Region in a defunct application heap, which could cause all sorts of problems for applications which were still running.

This version does not change the Handle, but rather modifies the contents of GrayRgn. If you crash with the Menu Bar hidden, GrayRgn is left pointing to a valid area of storage, at least. There are still problems since the Menu Bar can be left hidden, but these are slightly less severe than leaving a dangling Handle in the system heap.

Earle

```
{
This unit provides the ability to hide the Menu Bar, and to show
it.  When the Menu Bar is hidden, windows may be placed in the
area normally used for the Menu Bar.

Usage:
        PROCEDURE MBar_Init:
          Used to initialize global variables used here.
          Call once at beginning of application code.
        PROCEDURE MBar_be_Gone;
          Hides the Menu Bar.
        PROCEDURE MBar_Restore;
          Shows the Menu Bar.  Call this whenever you are going
          to be put into the background.  Call before Exit.

The procedures in this unit will probably break under some future
release of the Macintosh Operating System, because they manipulate
the GrayRgn.  They do work under MultiFinder 6.1a2.  The most
serious warning I can give concerning use of these routines is
that you must never allow your application to be placed into the
background with the Menu Bar hidden.

Possible compatibility problems:

  Modifies lowmem MBarHeight
  Modifies contents of GrayRgn

This file is part of Earle R. Horton's private source code library.
Earle R. Horton assumes no responsibility for any damages arising
out of use of this source code for any purpose.  Earle R. Horton
places no restrictions on use of all or any part of this source code,
except that this paragraph may not be altered or removed.

Original Language:
   MPW Pascal, v. 2.0.2
Origination Date:
   March 20, 1989
Modifications:
   April 4, 1989 ERH
     First version changed lowmem GrayRgn to point to a Region in the
application heap while the Menu Bar was hidden.  This version copies the
new Region to GrayRgn.  If the application crashes with the Menu Bar
hidden, GrayRgn no longer points to part of the defunct application
heap.  There are still problems, however, because the Menu Bar is left
hidden and other applications cannot access it.
}
UNIT MenuBar;
  INTERFACE
```

```
    USES
      {$Load PasDump.dump}
      Memtypes, Quickdraw, OSIntf, Script, ToolIntf;

    PROCEDURE MBar_be_Gone;
    PROCEDURE MBar_Restore;
    PROCEDURE MBar_Init;

    PROCEDURE SetMBarHeight(newheight:integer);
    INLINE smPopStack2Word,smMBarHeight;        { move.w        (a7)+,$0BAA }

    VAR
      Real_MBar_Height: integer;        { Copy of lowmem MBarHeight }
      Save_Region:      RgnHandle;      { Copy of GrayRgn }
      Hidden_Flag:      Boolean;        { State info }
      MBar_Rect:        Rect;           { Rect in which MBar is drawn }

  IMPLEMENTATION

    PROCEDURE MBar_Init;
    BEGIN
      Hidden_FLag := false;
      Real_MBar_Height := GetMBarHeight;
      SetRect(MBar_Rect,
              screenBits.bounds.left,
              screenBits.bounds.top,
              screenBits.bounds.right,
              screenBits.bounds.top + Real_MBar_Height);
    END;
{
Make the Menu Bar go away under MultiFinder or UniFinder.
Since this procedure manipulates the Gray Region, future
compatibility is unknown.
}
    PROCEDURE MBar_be_Gone;
    VAR
      MBar_Region:      RgnHandle;
      theWindow:        WindowPeek;
      BEGIN
        IF not Hidden_Flag THEN
          BEGIN
            Hidden_Flag := true;
                                              { Get some Regions to work with}
            Save_Region := NewRgn;
            MBar_Region := NewRgn;
                                             { Set the Menu Bar height to zero}
            SetMBarHeight(0);
 }
            CopyRgn(GetGrayRgn,Save_Region);
               { Fix up GrayRgn to cover the old GrayRgn plus the Menu Bar Rect}
            RectRgn(MBar_Region,MBar_Rect);
            UnionRgn(GetGrayRgn,MBar_Region,GetGrayRgn);
                    { Paint and fix up visRgn for any windows with exposed area}
            theWindow := WindowPeek(FrontWindow);
            PaintOne(theWindow,MBar_Region);
            PaintBehind(theWindow,MBar_Region);
            CalcVis(theWindow);
            CalcVisBehind(theWindow,MBar_Region);
            DisposeRgn(MBar_Region);                        { Clean up, leave}
          END;
      END;
{
Restore the Menu Bar and GrayRgn to normality.
Call when app is put into background.
}
```

```
    PROCEDURE MBar_Restore;
    VAR
      theWindow:        WindowPeek;
    BEGIN
    IF Hidden_Flag THEN
      BEGIN
        Hidden_Flag := false;
                           { Restore to original }
        CopyRgn(Save_Region,GetGrayRgn);
                        { Restore Menu Bar height }
        SetMBarHeight(Real_MBar_Height);
                     { Fix up any covered windows }
        RectRgn(Save_Region,MBar_Rect);
        theWindow := WindowPeek(FrontWindow);
        CalcVis(theWindow);
        CalcVisBehind(theWindow,Save_Region);
        DisposeRgn(Save_Region);
                { Draw the Menu, get out of here }
        HiliteMenu(0);
        DrawMenuBar;
      END;
    END;
END.
```

BitMap Rotation in C (and support routines)

by Juri Munkki

I'm including two small Think C programs along with sources, project files and resources.

I tried to make it as easy as possible to include these programs with the usenet programmer's guide. The resource files are extremely simple. You only need one window resource and one picture resource. The windows should have id 1000 and they should be visible, but all the other parameters can be whatever you want. The PICTs should have id 1000 and can be just about anything.

The other program documents the internal region data format by providing the functionality of the BitmapToRegion call. This may actually be useful to those programmers who wish to have their programs running on machines without 32 bit QD, although I recommend using Apple's licenseable code instead of mine. In addition, understanding the region data format will allow programmers a better understanding of quickdraw algorithms.

The other program performs a function that is not available from Apple or from anywhere else that I know of. So far programs have had to have their own bitmap rotation routines. A 90 degree rotation routine was already included with the guide, but this routine allows free rotation of any bitmap. Current performance is limited by the drawing speed. To optimize this routine, draw into an offscreen bitmap using your own commands. If there are enough requests for this, I could write something to do it more efficiently. Using PaintRect or MoveTo/LineTo is definitely not the way to do it.

I hope you find these interesting.

[Juri's code lined up beutifully until it was imported into Word.]

{---------- Bit map Rotation ----------}

```
/*
>>   BitRot.c   Bitmap Rotation Algorithm Tester.
>>            Copyright (c)1990, Juri Munkki
>>            Permission to use is granted for noncommercial applications.
>>
>>   This program rotates a bitmap to any angle. With modifications, it
>>   can be used to scale as well as rotate.
>>
>>   The idea is to perform the inverse tranformation to the destination
>>   bitmap. The trick is to avoid multiplication in the main loop. This
>>   program also optimizes so that it doesn't copy any extra pixels.
>>
>>   The routine does a simplified flood fill to copy every pixel in the
>>   destination from the source. There are no guarantees that every pixel
>>   from the source is used, but every pixel in the destination is checked.
>>
>>   The program works it's way from near the center of the rotated rectangle.
>>   To illustrate:
>>          .           .           .           .
>>         / \         / \         /_\         /_\
>>        /   \       /   \       /___\       /___\
>>       /  .  \     /_____\     /_____\     /_____\
>>       \     /     \     /     \     /     \_____/
>>        \   /       \   /       \   /       \___/
>>         \ /         \ /         \ /         \_/
>>          '           '           '           '
>>
>>   The fill operation starts from the center of the rectangle and works it's
>>   way up and down. While on the way, the point may also move left or right
>>   depending on a displacement value that is calculated beforehand. The fill
>>   always ends at topmost and bottommost corners. To Try out how the fill works,
>>   use a relatively dark picture and/or add some delays in the plotting subroutine.
>>
>>   To compile:
>>      long is 32 bits
>>      fixed point numbers are 16+16 bit.
>>      integers are 16 bits.
```

```
>>      Needs a 'WIND' id 1000 window template resource.
>>      Needs a 'PICT' id 1000 as the picture to rotate.
>>
>>   Limitations: For extremely large bitmaps, there might be a problem with fixed
>>   point resolution. To improve resolution, use larger fixed point numbers or
>>   extended precision floating point.
*/

#define   PIC_ID   1000         /* Picture resource ID 1000 is used as the demo picture  */

/*   Prototypes:
*/
void   PictBit(BitMap *,int);    /*  Creates a bitmap and draws the picture into
it.         */
int      origpixel(long,long);   /*  Almost same as GetPixel, but optimized for our
purposes   */

typedef   struct        /*   A Handy structure that keeps track of coordinates      */
{                       /*   in the source and destination rectangles.            */
   int      x,y;        /*   Location on destination bitmap in integral coordinates   */
   long   xo,yo;        /*   Location on source bitmap in fractional coordinates      */
}   place;

long   maxx,maxy;       /*   Size of source bitmap as a fractional (16:16) number.   */

WindowPtr   mywind;     /*   Any port in a storm for drawing the rotated bitmap.      */
BitMap      mybits;     /*   Bitmap to hold the source bits.                        */
long      sinr,cosr;    /*   Sin and Cosine values.       65536==1.0                */

/*
>>   ScanLeftRight tries to go as far left in the destination as the source
>>   rectangle permits. While doing this, it copies the pixels from the source
>>   to the destination. The same thing is done from the center to the right.
*/
void   ScanLeftRight(start)
place   *start;
{
   place   left,right;

   left=right=*start;               /*   Starting point.         */
   if(origpixel(left.xo,left.yo))   /*   Copy starting point.   */
   {   PlotDot(left.x,left.y);
   }

   /*   Check source rectangle boundaries.                  */
   while(left.xo>=0 && left.yo>=0 && left.xo<maxx && left.yo<maxy)
   {   if(origpixel(left.xo,left.yo))
      {   PlotDot(left.x,left.y);   /*   Copy pixel.            */
      }
      left.x--;                 /*   Move left.            */
      left.xo-=cosr;            /*   Move within source      */
      left.yo-=sinr;
   }

   right.x++;                  /*   Move right.            */
   right.xo+=cosr;
   right.yo+=sinr;

   /*   Check source rectangle boundaries.                  */
   while(right.xo>=0 && right.yo>=0 && right.xo<maxx && right.yo<maxy)
   {   if(origpixel(right.xo,right.yo))
      {   PlotDot(right.x,right.y);
      }
      right.x++;               /*   Move right            */
      right.xo+=cosr;
      right.yo+=sinr;
```

```
   }
}
void   main()
{
   long      theangle;             /*   Angle of rotation.               */
   place      goup,godown;         /*   Coordinates.                     */
   int         disp,edgex;         /*   Displacement (see below).        */
   Point      mouse,oldmouse;      /*   mouse locations for test.        */

   InitGraf(&thePort);      InitCursor();
   InitFonts();         InitWindows();
   InitMenus();         TEInit();
   InitDialogs(0L);                /*   Start up managers.              */
   mywind=GetNewWindow(1000,0,-1); /*   Open up a window.               */
   SetPort(mywind);                /*   Draw in this new window.        */

   PictBit(&mybits,PIC_ID);      /*   Read the picture into a bitmap.         */
   SetupGetPixel();              /*   Prepare for fast read of bitmap.      */

   maxx=((long)mybits.bounds.right)<<16;   /*   Source boundaries are changed   */
   maxy=((long)mybits.bounds.bottom)<<16;   /*   into fixed point numbers      */

   while(!Button())            /*   Quit when button is down.            */
   {   GetMouse(&mouse);         /*   Find out mouse location.            */
      if(mouse.h!=oldmouse.h)      /*   Has horizontal position changed?      */
      {   oldmouse.h=mouse.h;
         theangle=mouse.h*1024L;   /*   Rotation angle from horizontal value.   */

         EraseRect(&mywind->portRect);   /*   Erase window contents.         */

         sinr=FracSin(-theangle)>>14;   /*   Sin for inverse rotation.      */
         cosr=FracCos(-theangle)>>14;   /*   Cosine for inverse rotation.   */

         goup.x=mywind->portRect.right/2; /*   Destination center x coordinate.*/
         goup.y=mywind->portRect.bottom/2;   /*   --   ''   --      y coordinate.*/
         goup.xo=(long)mybits.bounds.right<<15;   /*   Center of source bitmap.*/
         goup.yo=(long)mybits.bounds.bottom<<15;   /*   Center of source bitmap.*/

         godown=goup;            /*   copy center to "godown".         */

         /*   Adjust starting position according to rectangle size and angle.
         **   Basically we transform one corner of the rectangle to find out
         **   where the fill should end.
         */
         if(cosr*sinr>0) disp=(-mybits.bounds.right*cosr + mybits.bounds.bottom*sinr)>>17;
         else  disp=(-mybits.bounds.right * cosr - mybits.bounds.bottom * sinr) >> 17;

         if(sinr>0)      disp=-disp;

         ScanLeftRight(&goup);   /*   Copy first line to destination.         */

         edgex= (disp>0) ? disp : -disp;   /*   edgex=ABS(disp)               */

         /*   Go up until source rectangle bound is crossed.               */
         do
         {   while(goup.xo>=0 && goup.yo>=0 && goup.xo<maxx && goup.yo<maxy)
            {   ScanLeftRight(&goup);

               goup.y--;      /*   Go up.                            */
               goup.xo+=sinr;   /*   Move in source bitmap coordinates.      */
               goup.yo-=cosr;
            }

            if(disp>0)         /*   Stay inside bounds as long as possible.   */
            {   goup.x++;      /*   This is done by adjusting the location   */
               goup.xo+=cosr;   /*   of the fill.                     */
```

```
            goup.yo+=sinr;
         }
         else
         {   goup.x--;
            goup.xo-=cosr;
            goup.yo-=sinr;
         }
      }   while(edgex-- > 0);   /*   Adjust only as long as it is useful.   */

      edgex= (disp>0) ? disp : -disp;   /*   edgex=ABS(disp)               */

      godown.y++;            /*   Go down.                        */
      godown.xo-=sinr;
      godown.yo+=cosr;

      /*   Go down until source rectanlge bound is crossed.           */
      do
      {   while(godown.xo>=0 && godown.yo>=0 && godown.xo<maxx && godown.yo<maxy)
         {   ScanLeftRight(&godown);

            godown.y++;
            godown.xo-=sinr;
            godown.yo+=cosr;
         }

         if(disp<0)
         {   godown.x++;
            godown.xo+=cosr;
            godown.yo+=sinr;
         }
         else
         {   godown.x--;
            godown.xo-=cosr;
            godown.yo-=sinr;
         }
      }   while(edgex-- > 0);
   }
  }
}
```

{----------- Bit Support for rotation ----------}

```
/*
>>  BitSupport.c  Bitmap Rotation Algorithm Tester.
>>          Support routines for bitmap rotation
>>          Copyright ©1990, Juri Munkki
>>          Permission to use is granted for noncommercial applications.
>>
>>  Ugly routines to test and set pixel values. You should start by optimizing
>>  these routines, if you wish to increase the speed of this program. The
>>  PlotDot routine is the real bottleneck of this program. Origpixel is quite
>>  fast, since it doesn't use toolbox routines.
*/

extern  BitMap    mybits;
    Ptr      *index;

void  PlotDot(x,y)
int    x,y;
{
  Rect  foo;

  foo.left=x;
  foo.right=x+1;
  foo.top=y;
  foo.bottom=y+1;
```

```
  PaintRect(&foo);
}

int    origpixel(x,y)
long  x,y;
{
asm  {  move.w  y,D0
    asl.w  #2,D0
    move.l  index,A0
    move.l  0(A0,D0),A0
    move.w  x,D0
    move.w  D0,D1
    lsr.w  #3,D0
    add.w  D0,A0
    moveq.l  #7,D0
    and.w  D0,D1
    sub.w  D1,D0
    btst  D0,(A0)
    beq    @nothing
    moveq.l  #-1,D0
    return
@nothing
    clr.w  D0
    return
  }
/*  return BitTst(index[y>>16],x>>16);*/
}

void  SetupGetPixel()
{
  int    i;
  Ptr    base;

  index=(Ptr *)NewPtr(mybits.bounds.bottom*sizeof(long));
  base=mybits.baseAddr;
  for(i=0;i<mybits.bounds.bottom;i++)
  {  index[i]=base;
    base+=mybits.rowBytes;
  }
}

/*  PictBit reads a picture resource, creates
>>  a large enough bitmap and draws the picture
>>  into it. The Bits bitmap is supplied to the
>>  routine. Space for the actual bits is reserved
>>  with NewPtr. Be sure to deallocate it once
>>  it is no longer needed!
>>
>>  No error checking is made.
*/
void  PictBit(Bits,PictId)
BitMap  *Bits;
int    PictId;
{
  GrafPort  AnyPort;
  GrafPtr    SavedPort;
  PicHandle  ThePic;
  Rect    TempRect;
  long    RAMNeeded;

  GetPort(&SavedPort);
  OpenPort(&AnyPort);

  ThePic=(PicHandle)GetResource('PICT',PictId);
  TempRect=(*ThePic)->picFrame;
```

```
  OffsetRect(&TempRect,-TempRect.left,-TempRect.top);
  Bits->bounds=TempRect;

  Bits->rowBytes=((TempRect.right + 15) >> 4) << 1;  /*  Round to word boundary  */
  RAMNeeded=Bits->rowBytes*TempRect.bottom;      /*  Calculate RAM for bits  */
  Bits->baseAddr=NewPtr(RAMNeeded);

  SetPortBits(Bits);
  AnyPort.portRect=TempRect;
  RectRgn(AnyPort.visRgn,&TempRect);
  RectRgn(AnyPort.clipRgn,&TempRect);

  EraseRect(&TempRect);
  DrawPicture(ThePic,&TempRect);

  ReleaseResource(ThePic);
  ClosePort(&AnyPort);
  SetPort(SavedPort);
}
```

{---------- Bit map to region ----------}

```
/*
>>  BitRegion.c, 04/23/89                          <<
>>  My routine for converting a bitmap into a region.          <<
>>                                    <<
>>  Juri Munkki, jmunkki@kampi.hut.fi                  <<
>>  Senior Systems Analyst                           <<
>>  Helsinki University of Technology Computing Centre          <<
>>  Otakaari 1 U044A, SF02150 Espoo, Finland              <<
>>                                    <<
>>  This program is in the PUBLIC DOMAIN, but:               <<
>>    I would really like to join the NeXT registered developer    <<
>>    program. I returned the forms, but I haven't heard anything    <<
>>    from NeXT. Please help me, if you can affect their decision.  <<
>>                                    <<
>>  Known bug:  This program knows how to create regions larger than  <<
>>        32 KB. QD doesn't support anything larger than 32KB.  <<
*/

#define  PIC_ID  1000      /*  Resource ID of test picture    */
RgnHandle  BitRgn(BitMap *);  /*  Function prototype for BitRgn  */

/*  PictBit reads a picture resource, creates
>>  a large enough bitmap and draws the picture
>>  into it. The Bits bitmap is supplied to the
>>  routine. Space for the actual bits is reserved
>>  with NewPtr. Be sure to deallocate it once
>>  it is no longer needed!
>>
>>  No error checking is made.
*/
void  PictBit(Bits,PictId)
BitMap  *Bits;
int    PictId;
{
  GrafPort  AnyPort;
  GrafPtr    SavedPort;
  PicHandle  ThePic;
  Rect    TempRect;
  long    RAMNeeded;

  GetPort(&SavedPort);
  OpenPort(&AnyPort);

  ThePic=(PicHandle)GetResource('PICT',PictId);
```

```
  TempRect=(*ThePic)->picFrame;

  OffsetRect(&TempRect,-TempRect.left,-TempRect.top);
  Bits->bounds=TempRect;

  Bits->rowBytes=((TempRect.right + 15) >> 4) << 1;  /*  Round to word boundary  */
  RAMNeeded=Bits->rowBytes*TempRect.bottom;      /*  Calculate RAM for bits  */
  Bits->baseAddr=NewPtr(RAMNeeded);

  SetPortBits(Bits);
  AnyPort.portRect=TempRect;
  RectRgn(AnyPort.visRgn,&TempRect);
  RectRgn(AnyPort.clipRgn,&TempRect);

  EraseRect(&TempRect);
  DrawPicture(ThePic,&TempRect);

  ReleaseResource(ThePic);
  ClosePort(&AnyPort);
  SetPort(SavedPort);
}

/*
>>  Convert a bitmap into a region.
>>  The bitmap origin should be at the top left corner and it
>>  shouldn't be wider than 8192 pixels. You might want to add
>>  some error checks for weird or illegal bitmaps.
*/
RgnHandle  BitRgn(Bits)
BitMap    *Bits;
{
  register Handle  Target;        /*  This is where we write the region  */
  register short  *TargetP;      /*  Pointer to region data array    */
  register long  MaxTarget;      /*  Memory management stuff        */
  register long  CurTarget;      /*  Index into the region data array  */
  long      RowStart;      /*  Index of first x value on row    */
  long      TargetSize,RgnSize;  /*  Size in data words & bytes      */
  Rect      TempRect,RgnBounds;  /*  Temporary & region bounds rects    */
  BitMap      RowBitMap;      /*  Working bitmap with one row in it  */
  char      RowBitData[1024];  /*  Buffer for pixels above (8192 pix)  */
  int        i;          /*  Row counter in a "for" loop      */
  register int  x;          /*  Column counter in a "for" loop    */
  register int  pixelstatus;    /*  Flag is false if last pixel is white*/

  TargetSize=4096;          /*  Initial guess for final region size  */
  Target=NewHandle(TargetSize);    /*  Allocate initial data buffer    */
  if(Target==0)  return 0;      /*  Did we run out of RAM? 0=failure.  */
  TargetP=(short *)(*Target + 10);  /*  TargetP points to region data    */
  HLock(Target);            /*  We just derefenced target. Lock it.  */
  MaxTarget=(TargetSize-20)/2;    /*  A safe maximum value for our index  */
  CurTarget=0;            /*  Start with target index 0 (no data)  */

  /*  Set region bounds to nothing:                    */
  SetRect(&RgnBounds,32767,32767,-32767,-32767);

  /*  Set up a bitmap with a single line:                 */
  TempRect=Bits->bounds;      /*  Set up left & right bounds      */
  TempRect.top=0;          /*  Single row bitmap with top=0    */
  TempRect.bottom=1;        /*  Single row bitmap with bottom=1    */
  RowBitMap.bounds=TempRect;
  RowBitMap.baseAddr=RowBitData;
  RowBitMap.rowBytes=((TempRect.right + 15) >> 4) << 1;

  /*  Start out with the first line of the source bitmap          */
  CopyBits(Bits,&RowBitMap,&TempRect,&RowBitMap.bounds,srcCopy,0);
```

```
  for(i=Bits->bounds.bottom;i>=0;i--)      /*  For every line and more      */
  {  TargetP[CurTarget++]=TempRect.top;    /*  Row data starts with Y value  */
    RowStart=CurTarget;            /*  X values on row start here    */
    pixelstatus=0;              /*  Pixels outside bitmap are white  */
    for(x=Bits->bounds.left;x<Bits->bounds.right;x++)
    {  if((BitTst(RowBitData,x)!=0) != pixelstatus)  /*  Test for a change  */
      {  pixelstatus= !pixelstatus;          /*  Color changed    */
        TargetP[CurTarget++]=x;            /*  Record x coordinate  */
        if(CurTarget>=MaxTarget)          /*  Is the buffer full?  */
        {  TargetSize+=2048;            /*  Enlarge the buffer    */
          HUnlock(Target);            /*  Unlock to change size  */
          SetHandleSize(Target,TargetSize);    /*  Change the size      */
          if(MemErr)                /*  No success?        */
          {  DisposHandle(Target);        /*  Dispose of what we have  */
            return 0;              /*  return failure.      */
          }
          TargetP=(short *)(*Target + 10);    /*  Dereference handle    */
          HLock(Target);              /*  Lock it again      */
          MaxTarget=(TargetSize-20)/2;      /*  New maximum index    */
        }
      }
    }
    if(pixelstatus)    TargetP[CurTarget++]=x;  /*  Last pixel was black, record edge  */
    if(RowStart==CurTarget)            /*  Row was empty (no changes)      */
      CurTarget--;              /*  Remove Y value from data      */
    else
    {  /*  Check for new region bounds:                        */
      if(TargetP[RowStart]   <RgnBounds.left)    RgnBounds.left=TargetP[RowStart];
      if(TargetP[CurTarget-1]>RgnBounds.right)  RgnBounds.right=TargetP[CurTarget-1];

      RgnBounds.bottom=TempRect.top;

      TargetP[CurTarget++]=32767;        /*  Write an "end of line" flag      */
    }

    /*  Copy current line into the single line bitmap:              */
    if(i>0) CopyBits(Bits,&RowBitMap,&TempRect,&RowBitMap.bounds,srcCopy,0);

    TempRect.top++; TempRect.bottom++;      /*  Move one line down      */

    /*  If we are still inside the bitmap, XOR this line with the previous line:*/
    if(i>1)  CopyBits(Bits,&RowBitMap,&TempRect,&RowBitMap.bounds,srcXor,0);
  }

  RgnBounds.top=TargetP[0];  /*  Top boundary is first recorded Y coordinate    */

  /*  If the region is empty, set the bounds rect to an empty rectangle:      */
  if(RgnBounds.right<=RgnBounds.left || RgnBounds.bottom<=RgnBounds.top)
    SetRect(&RgnBounds,0,0,0,0);

  TargetP[CurTarget++]=32767;  /*  Write an "end of region" flag          */
  HUnlock(Target);      /*  Unlock our target region.            */
  RgnSize=CurTarget*2+10;    /*  Calculate region size.              */
  if(RgnSize<=28)  RgnSize=10;  /*  Rectangular or empty region is only a Rect    */

  (*(RgnHandle)Target)->rgnBBox=RgnBounds;/*  Store region bounds rectangle    */
  (*(RgnHandle)Target)->rgnSize=RgnSize;  /*  Store region size (low 16 bits)    */
  SetHandleSize(Target,RgnSize);      /*  Resize region to optimally small  */
  return (RgnHandle)Target;        /*  Return resulting region handle    */
}

/*  This is just a short test program "main":
*/
void  main()
{
  WindowPtr  TestWindow;    /*  Simple window used for testing  */
```

```
  RgnHandle  TheRegion;    /*  Region handle for test region  */
  BitMap    TheBits;    /*  Bitmap for testing        */
  EventRecord  MyEvent;

  /*  "Magic Incantations" (Copyfight Apple Computer, Inc.)  */
  InitGraf(&thePort);    InitCursor();  InitFonts();      InitWindows();
  InitMenus();      TEInit();    InitDialogs(0L);    InitCursor();

  TestWindow=GetNewWindow(1000,0,-1);
  SetPort(TestWindow);

  PictBit(&TheBits,PIC_ID);  /*  Read the picture into a bitmap      */
  TheRegion=BitRgn(&TheBits);  /*  Convert the bitmap into a region    */
  if(TheRegion)        /*  If we get a region, let's play with it  */
  {  InvertRgn(TheRegion);  /*  Display region              */
    FlushEvents(everyEvent,0);
    while(GetNextEvent(mDownMask,&MyEvent)==0);

    GlobalToLocal(&MyEvent.where);
    InvertRgn(TheRegion);  /*  Hide region, then drag it around.    */
    DragGrayRgn(TheRegion,MyEvent.where,
          &TestWindow->portRect,
          &TestWindow->portRect,
          noConstraint,0L);
  }
}
```

INIT Skeleton Code

by Jon Wätte

SetWindow INIT, which lets you place windows anywhere on the screen when an application calls ShowWIndow on it. (Just like TWM under X)

The INIT consists of a loader (that should be compiled as an INIT resource) and two patches (of which the loader will choose to install one depending on color QD availability)

The patches should be compiled as "tpat" code resources, and the b/w version should have resource id 128, the color version id 129.

Stuff the three resources (INIT, and 2 tpat's) into a file of type INIT, and drop it into your system folder. Reboot and enjoy !

(Actually, this INIT should check for the option key being down before wanting to place a window, that would make it much more useful)

It is tested on a SE/30 with 24bit color and 32bit QD, and on a plain 1meg SE, and both works fine (The SE hasn't Color QD, and thus uses the b/w version)

Happy hacking,

Jon Wätte, Stockholm, Sweden, h+@nada.kth.se

```
/*

    SetWindow.c

    This INIT loads the tpat resource ID 128 for B/W and tpat 129 for color
    systems and patches the ShowWindow trap with that resource.

    Copyright 1990 Jon Wätte. Permission granted to use and distribute if
    you don't charge anything for it. If you do, a quarter of your gross
    sales is mine.
*/


int
strcmp(char * s1, char * s2) /* Since we don't want to link with the ANSI
                             library for just one functino, we do it
                             ourselves. Note, this verision returns 0 on
                             MISmatch ! */
{
    while(*s1 == *s2 && *s2) {
        s1++; s2++;
    }
    if(*s1 == *s2) return 1;
    return 0;
}


main()
{
    char * moof;
    long oldaddr;
    Handle foom;
    int num = 128;
    SysEnvRec theWorld;

    SysEnvirons(2, &theWorld); /* Check what we have here */
    if(theWorld.hasColorQD) num++; /* If we have color QD, use the CQD version
                                      which has another number, and supports
```

```
                                                multiple screens */
    foom = GetResource('tpat', num);

    if(foom == 0) { /* Maybe we built the resources with the wrong number ? */
        SysBeep(30); /* Beep to show we didn't load */
    } else {
        HUnlock(foom); /* May be marked as "locked" */
        DetachResource(foom); /* We don't want a resource hanging around in the
                                 system heap in that way... */
        MoveHHi(foom); /* Get the code as much out of the way as possible */
        HLock(foom); /* Lock it down firmly, so it won't move... */

        for(moof = *foom; !strcmp(moof, "Moof!"); moof++); /* Check for the
                                 availability of our "signature" */

        * (long *) moof = NGetTrapAddress(0x115, ToolTrap); /* Save the address
                                 to jump to in place of the signature */
        NSetTrapAddress(StripAddress(*foom), 0x115, ToolTrap); /* Set the new
                                 trap address to our routine - note, since the
                                 machine possibly might be SwapMMU'ed, we do a
                                 StrpAddress - it can't hurt anyway */
    } /* That's it ! Not so hard at all. */
}
```

{---------- Set window tpat ----------}

```
/*
    SetWindow tpat

    This trap patch will make ShowWindow act like X-windows,
    so you may place a new window wherever you like.

    This INIT does lots of stupid things, like pokes in lo-mem globals
    not very well-documented, and draws in the WMgrPort.

    Copyright 1990 Jon Wätte - distribution and usage allowed if you
    don't charge for it. If you do, quarter of your gross sales is mine.

    I'm reachable as Internet: h+@nada.kth.se USEnet: mcsun!sunic!draken!h+

*/

/*
    Things on the to-do list: Maybe turn on/off various features with a
    control panel cdev ? Maybe the mouse should move back again after
    positioning the window ? Maybe we shouldn't beep at dumb applications ?
    Maybe we should show Modal windows without positioning them ?
*/

/* lo-mem globals that are documented, somewhere... */
extern  Point   MTemp           :   0x828;
extern  Point   RawMouse        :   0x82c;
extern  int     CrsrNewCouple   :   0x8ce;

main(WindowPtr w) /* This is the patch. The declaration should look the same
                     as if you were writing the actual routine. Note, that for
                     routines taking more than one argument, pascal declaration
                     is needed. */
{
    char blackPat[8];

    int ofx = w->portRect.right - w->portRect.left,
        ofy = w->portRect.bottom - w->portRect.top; /* Calculate the dimensions
                                                                   of the window */
    int x;
```

```
    asm {
           bra        @done
    moof:  dc     'Mo', 'of', '!\000' /* This is used for communication
                                          with the loader, to see where to
                                          jump next */
    done:  nop
    }

/* Do the stuff here ! */

    for(x=0; x < 8; x++) blackPat[x] = 0x55 << (x & 1); /* Set up a grey
                                                                    pattern */
    if(!(((WindowPeek) w)->visible)) { /* Only if you show a hidden window */

        Point p; /* We have to save away various data to get the thing to
                    work right, and reset the WMgrPort in its state. Otherwise,
                    the various managers would get VERY confured */
        GrafPtr oldPort;
        GrafPtr myPort;
        PenState pnState;
        RgnHandle clipRgn = NewRgn();

        GetPort(&oldPort); /* Whatever port was used - note, usually apps
                              do a SetPort after a ShowWindow, but it never
                              hurts to be nice */
        GetWMgrPort(&myPort); /* This is the port we're gonna draw in */
        SetPort(myPort);

        GetClip(clipRgn); /* We have to change the clip region so we're sure
                             that drawing actually takes place */
        ClipRect(&(myPort->portRect)); /* No way of knowing more than one
                             monitor without Color QuickDraw */

        p = * (Point *) &(w->portBits.bounds); /* Where to move the mouse */
        p.h = -p.h; /* The offsets are negative in bounds ... */
        p.v = -p.v;
         RawMouse = p; /* Hit the mouse lo-mem globals (danger !!!) */
        MTemp = p;
        CrsrNewCouple = 0xffff;

        do {
            long l;
            Delay(2, &l); /* Give the mouse a chance to catch up to the place
                             where the window's default position is */
        } while(Button()); /* See to it the button's up before we go
                              further */

        GetPenState(&pnState);
        HideCursor(); /* We don't want the cursor obscured */
        ShowPen();
        PenSize(2, 2);
        PenMode(patXor); /* Drawing in the WMgrPort requires undo-able
                            ilnes only obtainable by xoring */
        PenPat(blackPat); /* it's really a grey pattern... */

        while(!Button()) {
            Rect r;
            long l;

            GetMouse((Point *) &r);
            r.bottom = r.top + ofy;
            r.right = r.left + ofx;
            FrameRect(&r); /* Show the outline */
            Delay(2, &l); /* For a short while */
            FrameRect(&r); /* Restore the screen */
        }; /* Until the user clicks */
```

```
        {
            GetMouse(&p); /* Where did the click go down ? */
            LocalToGlobal(&p);
            MoveWindow((WindowPtr) w, (int) p.h, (int) p.v, (Boolean) 0);
                /* Move the window we're placeing there */
        }

        ShowCursor(); /* Restore the saved state of the WMgrPort */
        HidePen();
        SetPenState(&pnState);
        SetClip(clipRgn);
        DisposHandle(clipRgn); /* Don't eat space, either... */

        SetPort(oldPort);
    } else {
        SysBeep(30); /* Here we beep if an application tries to show an
                        already visible window. Good for tracking unnecessary
                        ShowWindows */
    }

    asm {
            move.l  @moof, a0 /* The loader looks for "Moof!", and stores
                                 the place to jump to there */
            unlk    a6 /* ONLY if you use local variables ! */
            jmp         (a0) /* JMP, not JSR. This is not a tail patch, and thus,
                                shouldn't break any OS patch */
    }
}
```

{ ---------- Set window tpat Color ---------- }

```
/*
    SetWindow tpat

    This trap patch will make ShowWindow act like X-windows,
    so you may place a new window wherever you like.

*/

/* lo-mem globals that are documented, somewhere... */
extern  Point   MTemp           :   0x828;
extern  Point   RawMouse        :   0x82c;
extern  int     CrsrNewCouple   :   0x8ce;

main(CWindowPtr w)
{
    char blackPat[8];
    RGBColor savedC;

    int ofx = w->portRect.right - w->portRect.left,
        ofy = w->portRect.bottom - w->portRect.top;
    int x;

    asm {
            bra         @done
    moof:   dc      'Mo', 'of', '!\000'
    done:   nop
    }

/* Do the stuff here ! */

    for(x=0; x < 8; x++) blackPat[x] = 0x55 << (x & 1);
    if(!(((WindowPeek) w)->visible)) {
        Point p;
```

```
        GrafPtr oldPort;
        GrafPtr myPort;
        RGBColor c = { 0, 0, 0 };
        PenState pnState;
        RgnHandle clipRgn = NewRgn();

        GetPort(&oldPort);
        GetCWMgrPort(&myPort);
        SetPort(myPort);
        GetForeColor(&savedC);

        GetClip(clipRgn);
        SetClip(GetGrayRgn());

        if((w->portVersion & 0xE000) == 0) { /* If an old-style graf port */
            p = * (Point *) &(((GrafPtr) w)->portBits.bounds);
        } else {
            p = * (Point *) &((*(w->portPixMap))->bounds);
        }
        p.h = -p.h;
        p.v = -p.v;
         RawMouse = p;
        MTemp = p;
        CrsrNewCouple = 0xffff;

        do {
            long l;
            Delay(2, &l);
        } while(Button());

        GetPenState(&pnState);
        HideCursor();
        ShowPen();
        PenSize(2, 2);
        RGBForeColor(&c);
        PenMode(patXor);
        PenPat(blackPat);

        while(!Button()) {
            Rect r;
            long l;

            GetMouse((Point *) &r);
            r.bottom = r.top + ofy;
            r.right = r.left + ofx;
            FrameRect(&r);
            Delay(2, &l);
            FrameRect(&r);
        };

        {
            GetMouse(&p);
            LocalToGlobal(&p);
            MoveWindow((WindowPtr) w, (int) p.h, (int) p.v, (Boolean) 0);
        }

        ShowCursor();
        HidePen();
        SetPenState(&pnState);
        SetClip(clipRgn);
        DisposHandle(clipRgn);
        RGBForeColor(&savedC);

        SetPort(oldPort);
    } else {
        SysBeep(30);
```

```
    }

    asm {
            move.l  @moof, a0
            unlk    a6 /* ONLY if you use local variables ! */
            jmp         (a0)
    }
}
```

New Volume Scanning Algorithm

By John Norstad

In Tech Note #68, "Searching All Directories on an HFS Volume", Apple gives a very simple algorithm for disk scanning. There's a problem with this algorithm, however, which I discovered while working on my anti-virus program Disinfectant. I've come up with an improved algorithm that solves the problem. The new algorithm will be part of Disinfectant version 1.1, which we hope to release early next week.

I've wanted to "publish" this new algorithm so that everyone can benefit from it. comp.sys.mac.programmer seems as good a place as any!

Please understand that this problem is not a "bug" in Disinfectant 1.0, despite what MacWeek has to say :-) The "bug" is shared by any program which uses the TN 68 algorithm to do disk scanning, which I suspect is all programs which do disk scanning.

The basic idea outlined in Tech Note #68 is to make indexed calls to the PBGetCatInfo file manager routine. We'll use (abuse) the following notation for these calls:

```
r = PBGetCatInfo(d, i, o)
```

means "call PBGetCatInfo to get the i'th object o in directory d, with result code r." Note that r will be non-zero if there are no more objects in the directory.

The algorithm in TN 68, expressed in pseudo-c and stripped of all the bells and whistles, is as follows:

```
i = 1
while (true) {
   if (PBGetCatInfo(d, i, o)) break
   if o is a subdirectory call ourselves recursively to scan o
   if o is a file scan it
   i++
}
```

This algorithm seems quite simple and fool-proof at first glance, but it only works if you assume that no other users or tasks are creating or deleting files or directories while the scan is in progress.

As an extreme example, suppose we're scanning a server volume that contains two files named A and B and a directory C that contains another 1000 files. Suppose that while we're scanning file B some other user deletes file A. Our index i in the above algorithm is 2 while we're scanning file B. When we finish scanning file B we increment i to 3 and loop, calling PBGetCatInfo to get the third object in the directory. But there are now only two objects in the directory (B and C), so the PBGetCatInfo call returns a non-zero result code and we break out of the loop and quit. The net result is that we end up scanning only 2 out of the 1002 total files on the server!

This problem is most serious when scanning server volumes, where the probability of other users creating or deleting objects is often significant. The problem can also occur on local volumes under MultiFinder if other tasks are creating or deleting objects during a scan, or if our program itself creates or deletes objects on the volume during the scan. (Disinfectant 1.0 suffers from all three problems, but only the server problem is really serious.)

My solution is quite simple. I simply recall PBGetCatInfo immediately after scanning an object to see if it has changed its position in the directory. If the position has changed, I rescan the directory to attempt to locate the new position.

The revised algorithm is:

```
i = 1
while (true) {
   if (PBGetCatInfo(d, i, o)) break
   if o is a subdirectory call ourselves recursively to scan o
   if o is a file scan it
   n = the name of object o
```

```
      if (!PBGetCatInfo(d, i, o)) {  /* recall PBGetCatInfo */
         m = the name of object o
         if (n == m) {              /* usual case - no position change */
            i++                     /* continue scan with next object */
            continue
         }
      }
      oldi = i                      /* save our old location */
      i = 1                         /* start looking for our new location */
      while (true) {
         if (PBGetCatInfo(d, i, o)) {
            i = oldi                /* just in case we've been deleted in
            break                   /* the last few milliseconds */
         }
         m = the name of object o
         if (n == m) {              /* found new location */
            i++                     /* continue scan with next object */
            break
         }
         i++
      }
   }
```

There is still an unavoidable window in this algorithm where our PBGetCatInfo indices can get out of synch with reality, but it is now only milliseconds wide instead of seconds or even minutes wide. So the new algorithm is still not perfect, but it's orders of magnitude better than the old naive one.

In my first attempt to design this new algorithm I tried to be fancy - I didn't rescan from the beginning of the directory, but I instead tried to scan backwards or forwards from the current position. This technique was slightly faster, but assumed that the directory was maintained in alphabetical order using the RelString toolbox routine with caseSens=false and diacSens=true. This works OK on normal volumes, but with foreign file systems and in other "non-standard" cases we can't assume that directories are in any particular order. The final algorithm presented above does not depend on directories being maintained in any particular order.

Please note that my new algorithm hasn't yet been put to the acid test of use by millions of real live users. But I think it's reasonable and it has worked just fine in my tests. Apple, of course, knows nothing about all this. If they did they'd probably tell me that it would break in system 7.0 :-) So use it at your own risk, etc., etc.

It's interesting that this problem is not shared by UNIX and other operating systems. In UNIX once an entry is made in a directory its position never changes. When entries are deleted they're simply marked "unused". The system does not attempt to move all the following entries down to close up the hole. There is no attempt made to keep the directories in any particular order.

The new algorithm is part of the reusable module scan.c, which is part of the "public" source code of Disinfectant. Write to me at the address below if you'd like a copy.

Please excuse the length of this posting. I thought this was a nifty trick, and there might be others who will find it useful.

John Norstad
Academic Computing and Network Services
Northwestern University

Bitnet: jln@nuacc
Internet: jln@acns.nwu.edu
AppleLink: a0173
CompuServe: 76666,573

DEC	CHAR	HEX	CNTL[1]
00	□NUL	00	^@
01	SOH	01	^A
02	STX	02	^B
03	ETX	03	^C
04	EOT	04	^D
05	ENQ	05	^E
06	ACK	06	^F
07	BEL	07	^G
08	BS	08	^H
09	HT	09	^I
10	LF	0A	^J
11	VT	0B	^K
12	FF	0C	^L
13	CR	0D	^M
14	SO	0E	^N
15	SI	0F	^O
16	DLE	10	^P
17	DC1	11	^Q
18	DC2	12	^R
19	DC3	13	^S
20	DC4	14	^T
21	NAK	15	^U
22	SYN	16	^V
23	ETB	17	^W
24	CAB	18	^X
25	EM	19	^Y
26	SUB	1A	^Z
27	ESC	1B	^[
28	FS	1C	^\
29	GS	1D	^]
30	RS	1E	^^
31	US	1F	^_
32	Space	20	
33	!	21	
34	"	22	
35	#	23	
36	$	24	
37	%	25	
38	&	26	
39	'	27	
40	(	28	
41	)	29	
42	*	2A	
43	+	2B	
44	,	2C	
45	-	2D	
46	.	2E	
47	/	2F	
48	0	30	
49	1	31	
50	2	32	
51	3	33	
52	4	34	
53	5	35	
54	6	36	
55	7	37	
56	8	38	
57	9	39	
58	:	3A	
59	;	3B	
60	<	3C	
61	=	3D	
62	>	3E	
63	?	3F	
64	@	40	
65	A	41	
66	B	42	
67	C	43	
68	D	44	
69	E	45	
70	F	46	
71	G	47	
72	H	48	
73	I	49	
74	J	4A	
75	K	4B	
76	L	4C	
77	M	4D	
78	N	4E	
79	O	4F	
80	P	50	
81	Q	51	
82	R	52	
83	S	53	
84	T	54	
85	U	55	
86	V	56	
87	W	57	
88	X	58	
89	Y	59	
90	Z	5A	
91	[	5B	
92	\	5C	
93	]	5D	
94	^	5E	
95	_	5F	
96	`	60	
97	a	61	
98	b	62	
99	c	63	
100	d	64	
101	e	65	
102	f	66	
103	g	67	
104	h	68	
105	i	69	
106	j	6A	
107	k	6B	
108	l	6C	
109	m	6D	

[1] The CNTL column tells you which key to use to get that ASCII value. The circumflex ^ represents the control key.

DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR	HEX
110	n	6E	169	©	A9	228	‰	E4
111	o	6F	170	™	AA	229	Â	E5
112	p	70	171	´	AB	230	Ê	E6
113	q	71	172	¨	AC	231	Á	E7
114	r	72	173	≠	AD	232	Ë	E8
115	s	73	174	Æ	AE	233	È	E9
116	t	74	175	Ø	AF	234	Í	EA
117	u	75	176	∞	B0	235	Î	EB
118	v	76	177	±	B1	236	Ï	EC
119	w	77	178	≤	B2	237	Ì	ED
120	x	78	179	≥	B3	238	Ó	EE
121	y	79	180	¥	B4	239	Ô	EF
122	z	7A	181	µ	B5	240		F0
123	{	7B	182	∂	B6	241	Ò	F1
124	l	7C	183	∑	B7	242	Ú	F2
125	}	7D	184	∏	B8	243	Û	F3
126	~	7E	185	π	B9	244	Ù	F4
127	□	7F	186	∫	BA	245	ı	F5
128	Ä	80	187	ª	BB	246	ˆ	F6
129	Å	81	188	º	BC	247	˜	F7
130	Ç	82	189	Ω	BD	248	¯	F8
131	É	83	190	æ	BE	249	˘	F9
132	Ñ	84	191	ø	BF	250	˙	FA
133	Ö	85	192	¿	C0	251	°	FB
134	Ü	86	193	¡	C1	252		FC
135	á	87	194	¬	C2	253	˝	FD
136	à	88	195	√	C3	254		FE
137	â	89	196	ƒ	C4	255	DEL	FF
138	ä	8A	197	≈	C5			
139	ã	8B	198	∆	C6			
140	å	8C	199	«	C7			
141	ç	8D	200	»	C8			
142	é	8E	201	…	C9			
143	è	8F	202		CA			
144	ê	90	203	À	CB			
145	ë	91	204	Ã	CC			
146	í	92	205	Õ	CD			
147	ì	93	206	Œ	CE			
148	î	94	207	œ	CF			
149	ï	95	208	–	D0			
150	ñ	96	209	—	D1			
151	ó	97	210	“	D2			
152	ò	98	211	”	D3			
153	ô	99	212	‘	D4			
154	ö	9A	213	’	D5			
155	õ	9B	214	÷	D6			
156	ú	9C	215	◊	D7			
157	ù	9D	216	ÿ	D8			
158	û	9E	217	Ÿ	D9			
159	ü	9F	218	⁄	DA			
160	†	A0	219	€	DB			
161	°	A1	220	‹	DC			
162	¢	A2	221	›	DD			
163	£	A3	222	fi	DE			
164	§	A4	223	fl	DF			
165	•	A5	224	‡	E0			
166	¶	A6	225	·	E1			
167	ß	A7	226	,	E2			
168	®	A8	227	„	E3			

```
Memory Manager
  MemFullErr    = - 108; { Not enough room in heap zone }
  NilHandleErr  = - 109; { Master Pointer was NIL in HandleZone or other }
  MemWZErr      = - 111; { WhichZone failed (applied to free block) }
  MemPurErr     = - 112; { trying to purge a locked or non-purgeable block }
  MemLockedErr  = - 117; { Block is locked }

Print Manager
  iMemFullErr   = - 108;
  iPrAbort      = 128;
  iIOAbort      = - 27;

Resource Manager
  resNotFound   = - 192; { Resource not found }
  resFNotFound  = - 193; { Resource file not found }
  addResFailed  = - 194; { AddResource failed }
  rmvResFailed  = - 196; { RmveResource failed }
  resAttrErr    = -198;  {attribute does not permit operation}
  mapReadErr    = -199;  {map does not permit operation}

File Manager
  DirFulErr     = - 33; { Directory full }
  DskFulErr     = - 34; { disk full }
  NSVErr        = - 35; { no such volume }
  IOErr         = - 36; { I/O error }
  BdNamErr      = - 37; { bad name }
  FNOpnErr      = - 38; { File not open }
  EOFErr        = - 39; { End of file }
  PosErr        = - 40; { tried to position to before start of file (r/w) }
  MFulErr       = - 41; { memory full(open) or file won't fit (load) }
  TMFOErr       = - 42; { too many files open }
  FNFErr        = - 43; { File not found }

  WPrErr        = - 44; { diskette is write protected }
  FLckdErr      = - 45; { file is locked }
  VLckdErr      = - 46; { volume is locked }
  FBsyErr       = - 47; { File is busy (delete) }
  DupFNErr      = - 48; { duplicate filename (rename) }
  OpWrErr       = - 49; { file already open with with write permission }
  ParamErr      = - 50; { error in user parameter list }
  RFNumErr      = - 51; { refnum error }
  GFPErr        = - 52; { get file position error }
  VolOffLinErr  = - 53; { volume not on line error (was Ejected) }
  PermErr       = - 54; { permissions error (on file open) }
  VolOnLinErr   = - 55; { drive volume already on-line at MountVol }
  NSDrvErr      = - 56; { no such drive (tried to mount a bad drive num) }
  NoMacDskErr   = - 57; { not a mac diskette (sig bytes are wrong) }
  ExtFSErr      = - 58; { volume in question belongs to an external fs }
  FSRnErr       = - 59; { file system rename error }
  BadMDBErr     = - 60; { bad master directory block }
  WrPermErr     = - 61; { write permissions error }

  lastDskErr    = - 64; { last in a range of disk errors }
  noDriveErr    = - 64; { drive not installed }
  offLinErr     = - 65; { r/w requested for an off-line drive }
  noNybErr      = - 66; { couldn't find 5 nybbles in 200 tries }
  noAdrMkErr    = - 67; { couldn't find valid addr mark }
  dataVerErr    = - 68; { read verify compare failed }
  badCkSmErr    = - 69; { addr mark checksum didn't check }
  badBtSlpErr   = - 70; { bad addr mark bit slip nibbles }
  noDtaMkErr    = - 71; { couldn't find a data mark header }
  badDCkSum     = - 72; { bad data mark checksum }
  badDBtSlp     = - 73; { bad data mark bit slip nibbles }
  wrUnderRun    = - 74; { write underrun occurred }
  cantStepErr   = - 75; { step handshake failed }
```

```
tk0BadErr     = - 76; { track 0 detect doesn't change }
initIWMErr    = - 77; { unable to initialize IWM }
twoSideErr    = - 78; { tried to read 2nd side on a 1-sided drive }
spdAdjErr     = - 79; { unable to correctly adjust disk speed }
seekErr       = - 80; { track number wrong on address mark }
sectNFErr     = - 81; { sector number never found on a track }
firstDskErr   = - 84; { first in a range of disk errors }

DirNFErr      = - 120; { Directory not found }
TMWDOErr      = - 121; { No free WDCB available }
BadMovErr     = - 122; { Move into offspring error }
WrgVolTypErr  = - 123; { Wrong volume type error - operation not supported for MFS}
FSDSIntErr    = - 127; { Internal file system error }
```

Authors and Credits

Al Evans 90
Alan Mimms 202
Alan T Goates 69
Alastair Dallas 184
Alex CHAFFEE 50
Alex Kazim 85
Amanda Walker 19, 21, 112, 216
Andrew Shebanow 42, 124
Ari Halberstadt 4, 5, 162
Ben Cranston 18, 29, 65, 82, 236
Bill Hofmann 216
Bill Stackhouse 103
Brian Bechtel 83, 90
Brian Patrick Arnold 8
Byron Han 201
Chris Muir 10, 21
Colin Klipsch 42, 51, 209
Dan McMullen 132
Dave ? 141
David Dantowitz 33
David N. Schlesinger 107
David Phillip Oster 82, 118, 120, 121, 143, 145, 149, 151, 156, 158, 165, 205, 210, 211
David Stoms 86, 127, 208
Earle R. Horton 122, 184, 257
Ed Tecot 125
Ephraim Vishniac 166
Eric Olson 114
Eric Pepke 4, 215, 221
Forrest Tanaka 185
Glenn L. Austin 27, 30, 108, 150
Gregory Dudek 128
Hal Perkins 71
Harry Chesley 63, 100
James Borquist 64
Jeff Stearns 15
Jim Logan loganj@byuvax.bitnet 21
Jim Mann 8
Jim Reekes 201
Jim Walker 191
Joe Holt 46, 53
John Bruner 127
John Chew 2
John Doner 13
John Norstad 255,275
John Olsen 171
Jon Wätte 269
Josh N. Pritikin 59
Juri Munkki 163, 182, 212, 260
Keith Rollin 68, 70, 87, 95, 190
Kelesi 169
Krzysztof Kozminski 84
Larry Rosenstein 22, 30, 51, 140, 163, 186, 192, 203, 209, 243
Lawrence D'Oliveiro 28, 39, 73
Leonard Rosenthol 23, 97
Lloyd Lim 71, 125, 246
Lon Hildreth 190
Mahboud Zabetian 45
Marc T. Kaufman 32, 34, 187
Mark B. Johnson 200
Martin Minow 7, 77, 193, 197
Matthew Mashyna 96, 134
Matthew T. Russotto 38, 126, 188
Matthew Xavier Mora 23, 203, 241
Matthias Urlichs 53
Michael Nolan 34
Michael Odawa 129
Miguel Calejo 132
Mike Morton 171
Murat Konar 52, 215
Ned Horvath 72, 115, 164
Nick Jackiw 31, 38, 77, 78, 85, 88, 97, 112, 167, 169, 191
Patrick C Beard 12, 124, 163, 192
Paul Snively 79
Pete Gontier 12, 15
Pete Keleher 2
Pete Resnick 63
Pierce T. Wetter III 187
Ray Spalding 99
Raymond C. Fischer) 166
Reynir Hugason 29
Rich Siegel 121, 123
Richard Clark 28
Richard M. Siegel 134
Rick Fleischman 5
Rick Holzgrafe 62, 108, 118
Robert J Woodhead 157
Roger H. Brown 31
Scott A. Mason 40
Sean Parent 43, 44
Stephen D Hawley 41
Steve Brecher 100
Steve Dorner 9
Steve M. Hoffman 202
Ted Woodward 126
Tim Maroney 3, 7, 44, 91, 94, 98, 101, 107, 114, 157, 158, 200, 214
Tom Dowdy 12, 140, 192
Tom Lippincott 87, 120
unknown 2, 96, 171
Vladimir G. Ivanovic 35
Walt Leipold 58