

USENET

Macintosh Programmer's Guide

Volume I

Thursday, October 18, 1990
(Updated)

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 SetClickLoop?.....	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 SysEnviron).....	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 Preferred 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 answers.....	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
Ilci 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

This COMPLATION of USENET extracts and articles (not the USENET extracts and articles themselves) is copyrighted © 1990 by Matthew Xavier Mora. You are free to distribute this volume in anyway you see fit provided that you do not make a profit from doing so. Each individual article is copyrighted by the author who wrote said article. If you use any code that is contained in this volume, please indicate who it came from in your about box or documentation (*there is no law saying you have to, but it sure would be nice of you*). Any NON Profit or not for profit user groups (such as BMUG and SMUG) are here by granted to include this volume in their software library disks. Any organization who makes a profit by selling public domain and/or shareware software **MUST obtain permission from me to distribute this volume.**

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 SetClickLoop?.....	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@mounthn.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 and Cartwright Reed.

It 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 }
HAppParms = ^PAppParms;
PAppParms = ^RAppParms;
RAppParms = 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    : HAppParms;
  AppParmLoc    : LongPtr;
  fooVRefNum    : INTEGER;
BEGIN
  { don't ask - use SFGGetFile or another means }
  fooVRefNum := MagicIncantationForGettingFileVRefNums( 'Foo' );

  { Set appParms to launch and open files }
  myAppParms := HAppParms( NewHandle( SIZEOF( RAppParms ) ) );
  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 apparmhandle 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^;
    SetClickLoop(@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@yvox.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}
```

```

CrshrNew = $8ce;      {byte}
CrshrCouple = $8cf;   {byte}
Couple = $ff;         {value for CrshrCouple}
Uncouple = $00;       {value for CrshrCouple}
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(CrshrNew);
  LowMem^ := $ffff;          {both CrshrNew & CrshrCouple}
  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 the 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 it's just PICT's that you want, that's easy. If you 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 their 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 Rosenthal)
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;
{
    intiCount = 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     @lp
#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
    lp: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_ido@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 `GetDlgItem` 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 | Visual Geometry Project | 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 $\text{maxInt} = 2^{31} - 1$.

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

| Roger H. Brown GENie: R.BROWN64 AOL: Roger48 |

●●●

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-congruential random number generators (of the form: $\text{seed} = \text{seed} * a \bmod m + b$) 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 modulus (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 n_1 , n_2 , and n_3 . (Relatively prime means these numbers have no common factors above 1.)

Please pardon my pseudocode:

```
This_Random_Number = (Previous_Random_Number *  $n_1$  +  $n_2$ ) mod  $n_3$ .
```

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

This will generate a random number sequence of period n_3 . 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! Among 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 8.12544×10^{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