Chapter 6 File Manager

From: blob@apple.com (Brian Bechtel)
Subject: Re: Finding amount of free disk space?

In article <4995@uhccux.uhcc.hawaii.edu> mikem@uhccux.uhcc.hawaii.edu (Mike Morton) writes:
> I've got a document open which a user specified with SFGetFile, so all I
> have to identify it is a vRefNum and a name. I'd like to find out how
> much free space there is on the drive on which the document is stored
> (i.e., how much more can I expand the document?).
>
> It looks like I want to use GetVInfo, which takes a drvNum as a
> parameter. If so, how do I map a vRefNum to the drvNum containing it?
> If not, what's the right way?

Call PBHGetVInfo, passing an ioCompletion of NIL, ioNamePtr of NIL, ioVRefNum of the vRefNum gotten from SFGetFile, and ioVolIndex of 0. You'll get back ioVFrBlk and ioVAlBlkSiz. Multiply those together to get the free bytes on the disk. Or, if you're lazy, pass ioVDrvInfo to your high level call of GetVInfo, and you'll get the multiplication done for you.

--Brian Bechtel   blob@apple.com

●●●

From:Al Evans-
Subject: How to get pathname (with Code)

In article <850zebolskyd@yvax.byu.edu> zebolskyd@yvax.byu.edu writes:

>If you want the _user_ to be able to see the pathname (and find their way to
>the file) you can use PBGetWDInfo to get the dirID from your working
>directory, then PBGetCatInfo to get that directory's parent's dirID, and
>so on until to get to dirID=2, which will be the root. You get the names
                  ^^^^^^^
  Although this seems to be true, it doesn't seem to be documented :-(

>along the way from ioNamePtr. That's also how I get the volume name for
>the volumename/dirID/filename resource I use to save a file's location
>for future _machine_ use. I can send you Modula-2 source code if you want,
>but you would probably be better off making your _own_ mistakes instead
>of trying to find mine.
>
>--Lyle D. Gunderson   zebolskyd@yvax.byu.edu   CIS: 73760,2354

There are MANY good reasons for never using full pathnames in a Mac application. But I, too, have stumbled across situations where it was absolutely necessary. I found that the technique recommended above works, but was unable to find any guarantee that the root directory would ALWAYS have dirID=2. As far as I can tell, the following routine does not rely upon anything undocumented:

-----cut about here--------
```
{
Returns full pathname to folder specified by startID in thePath, where
startID is the vRefNum/wdRefNum returned by SFGetFile or SFPutFile
}

  PROCEDURE GetCurrentPath (startID : INTEGER; VAR thePath : Str255);
      VAR
          tempName : Str255;
          vParams : CInfoPBRec;
          theError : OSErr;
  BEGIN
      thePath := '';
      tempName := '';
      WITH vParams DO
```

```
BEGIN
    ioCompletion := NIL;
    ioNamePtr := @tempName;
```

```
              ioVRefNum := startID;
              ioFDirIndex := -1;
              ioDrDirID := 0;
              REPEAT
                  theError := PBGetCatInfo(@vParams, FALSE);
                  IF (theError = noErr) THEN
                    BEGIN
                        ioDRDirID := ioDRParID;
                        thePath := concat(tempName, ':', thePath);
                        tempName := '';
                    END;
              UNTIL (theError <> noErr);
          END;
    END;
```

--Al Evans--

●●●

From: Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com
Subject:User items in SFdialog

In article <2426@tekno.chalmers.se> d83_sven_a@tekno.chalmers.se (SVEN AXELSSON) writes:
>Hi.

Hello.

> >I want to put a user-item into my custom SFGetFile dialog. How do I get the
> >user-item procedure connected? I can't do the usual GetItem - SetItem stuff
> >since SFPGetFile will read the dialog template for me. Is there a way of
> >handling this, short of writing my own filterProc - something I do NOT
> >want to do.

No. But I don't see what's wrong with writing your own filter proc. Just type in the function definition from Inside Mac and pass the name of the function to SFPGetFile. It's certainly no harder than writing a user item drawing procedure.

Your item filter procedure will get passed a -1 item for initialization. This is the time to do the GetDItem-SetDItem soft-shoe to bind the user item drawing procedure. The entire function shouldn't take more than ten lines if this is all you're doing.
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: **SFGetFolder (with Code)**

In article <709@maytag.waterloo.edu> jb@aries5.UUCP () writes:
> >Has anybody written a SFGetFolder/SFPutFolder routine -- What I want to be
> > able to do is select or create a folder in a nice manner.

This is the second most common question here, after "How do I get a full file name?" Apple, are you listening? How about a couple of new traps in System 7.0?

Anyway, the answer is yes. LSC source code at end of message.

> >The problem that I have with the implementations that I have seen of
> >selecting a folder i.e. 'Set Transfer Directory...' in Telnet is that it
> >is not intuitive whether I am selecting the directory of the folder
> >that is hilighted or the directory that I am currently in.

File Manager

That's very perceptive. The solution is not to allow the user to click on the "Use Folder" (or whatever) button when a folder is selected, and to put a sentence in the dialog explaining which folder is being dealt with just to make sure there's no confusion. Hardly anyone does this in practice, but everyone should.

Here goes with the code.

```
static Boolean good, noSys, needWrite, allowFloppy, allowDesktop;
static SFReply reply;

pascal Boolean
FolderFilter(pb)
FileParam *pb;
{
        return true;
}

pascal short
FolderItems(item, dlog)
short item;
DialogPtr dlog;
{
        if (item == 2) {
                good = true;
                item = 3;
        }
        return item;
}

pascal void
FolderEvents(dialog, event, item)
DialogPtr dialog;
EventRecord *event;
short *item;
{
        ControlHandle ch;
        short type;
        Rect r;
        HVolumeParam vp;

        /* disable if a directory is selected in the list */
        GetDItem(dialog, 2, &type, &ch, &r);
        if (reply.fType) {
                HiliteControl(ch, 255);
                return;
        }

        /* get information on the volume */
        vp.ioNamePtr = (StringPtr)0;
        vp.ioVRefNum = -SFSaveDisk;
        vp.ioVolIndex = 0;
        if (PBHGetVInfo(&vp, false))
                HiliteControl(ch, 255);
        else if (vp.ioVSigWord != 0x4244)            /* HFS? */
                HiliteControl(ch, 255);
        else if (vp.ioVDRefNum >= 0 || vp.ioVDrvInfo == 0)  /* ejected? */
                HiliteControl(ch, 255);
        else if (needWrite && (vp.ioVAtrb & 0x8080))      /* locked? */
                HiliteControl(ch, 255);
        else if (!allowFloppy && vp.ioVDRefNum == -5)     /* floppy? */
                HiliteControl(ch, 255);
        else if (!allowDesktop && CurDirStore == 2)       /* desktop? */
                HiliteControl(ch, 255);
        else if (noSys && CurDirStore == vp.ioVFndrInfo[0]) /* blessed? */
```

```
            HiliteControl(ch, 255);
    else    HiliteControl(ch, 0);
}
```

```
GetFolder(name, volume, folder, writeable, system, floppy, desktop)
char *name;
short *volume;
long *folder;
Boolean writeable, system, floppy, desktop;
{
        short oldvol = -SFSaveDisk;
        long oldfolder = CurDirStore;
        Point where;
        SetPt(&where, 55, 55);
        good = false;
        if (*volume && *folder) {
                SFSaveDisk = -*volume;
                CurDirStore = *folder;
        }
        needWrite = writeable, noSys = !system, allowFloppy = floppy;
        allowDesktop = desktop;
        SFPGetFile(where, (char *)0, FolderFilter, -1, 0, FolderItems,
                &reply, 14, FolderEvents);
        if (!good) return false;
        *volume = -SFSaveDisk;
        *folder = CurDirStore;
        FullFileName(name, "", -SFSaveDisk, CurDirStore);
        SFSaveDisk = -oldvol;
        CurDirStore = oldfolder;
        return true;
}
```

Note -- you need dialog 14 for this to work. This is a slightly hacked
version of Standard File. Here's a Rez listing made by DeRez:

```
resource 'DLOG' (14, "Standard File for a Folder") {
    {55, 78, 312, 439},
    dBoxProc,
    invisible,
    noGoAway,
    0x0,
    14,
    ""
};

resource 'DITL' (14, "Standard File for a Folder") {
    {   /* array DITLarray: 11 elements */
        /* [1] */
        {33, 507, 51, 587},
        Button {
            enabled,
            "Open"
        },
        /* [2] */
        {135, 256, 153, 336},
        Button {
            enabled,
            "Use Folder"
        },
        /* [3] */
        {161, 256, 179, 336},
        Button {
            enabled,
            "Cancel"
        },
        /* [4] */
```

File Manager

```
{40, 247, 62, 348},
UserItem {
    disabled
```

```
        },
        /* [5] */
        {69, 256, 87, 336},
        Button {
            enabled,
            "Eject"
        },
        /* [6] */
        {95, 256, 113, 336},
        Button {
            enabled,
            "Drive"
        },
        /* [7] */
        {40, 15, 185, 246},
        UserItem {
            enabled
        },
        /* [8] */
        {40, 229, 185, 246},
        UserItem {
            enabled
        },
        /* [9] */
        {124, 251, 125, 339},
        UserItem {
            disabled
        },
        /* [10] */
        {97, 606, 198, 702},
        StaticText {
            disabled,
            ""
        },
        /* [11] */
        {196, 15, 246, 345},
        StaticText {
            disabled,
            "Move until ^0 is shown in the small rect"
            "angle at the top, above the list of file"
            "s and folders.  Then click \"Use Folder\"."
        }
    }
};
```

The "^0" is filled in by the caller of GetFolder, using ParamText. Granted, that's not the cleanest solution, but GetFolder already has too many parameters. If you are only using this for one thing in your software, then you can always just fill it in in the dialog item in the resource file instead.

No flames from symbolic-constant freaks. It's just as easy to change a number in a routine as it is to change it in #defines, if it only appears once. And the dialog items are not going to change; their order is mandated by Standard File.

I do apologize for the use of globals, but as all Standard File hackers know, the system doesn't give you the ability to do without them easily. If your filter procs were passed a pointer to the reply record, you could embed it in a structure and use its pointer as a structure pointer, but it doesn't get passed. And touching the dialog refCon causes an explosion. If you really can't cope with globals, you're stuck with using a button refCon or something, and that's a bit of a mess. (Also, I don't think I'd figured out that kind of devious solution back when I wrote this.)

The one great omission in this code is a "New" button. One day I'll get around to adding it.
--

●●●

From: tim@hoptoad.uucp (Tim Maroney)

Subject: Re: Repeat SetVol query

From article <3260@carthage.cs.swarthmore.edu>, by jackiw@cs.swarthmore.edu (Nick Jackiw):
> You can't SetVol(...) to the working directory returned by SFGetFile
> under HFS.

In article <2802@hub.UUCP> 6600pete@hub.UUCP writes:
> Sure you can. I've done it lots. What error code are you getting back?

In article <3273@carthage.cs.swarthmore.edu> jackiw@carthage (Nick Jackiw) writes:
>No error. Just no results. Seems to me the following program, which
>calls SFGetFile thrice, should on the last instance bring up the GetFile
>dialog in the same directory/volume as a file was chosen in on the first
>instance. Got that?

Ah-hah! This once again goes to show how important it is to list your symptoms in detail. If you'd said this, you would have been besieged by messages pointing out that Standard File and the current HFS directory have almost nothing to do with each other.

If you have to set the directory for Standard File (and this *is* sometimes a valid operation, though indiscriminate use can confuse the user -- remember that power users in particularly frequently "blind type" to Standard File and expect it to work predictably) you do it
with a pair of low-memory globals called SFSaveDisk (negative of volume reference number) and CurDirStore (directory id). I demonstrated this technique in the code I just posted on hacking Standard File to select folders. It's more or less documented in Inside Mac IV, chapter 15. Set these globals to the necessary values before your call to Standard File.

To do this with a working directory reference number, like the one you get from Standard File, you have to convert the working directory into a volume/folder pair. You do this like so:

```
SFVolDir(wd, volume, folder)
short wd, *volume;
long *folder;
{
       WDPBRec pb;
       char name[72];
       pb.ioNamePtr = (StringPtr)name;
       pb.ioVRefNum = wd;
       pb.ioWDIndex = 0;
       pb.ioWDProcID = 0;
       pb.ioWDVRefNum = 0;
       PBGetWDInfo(&pb, false);
       if (pb.ioResult) return pb.ioResult;
       *volume = pb.ioWDVRefNum;
       *folder = pb.ioWDDirID;
       return noErr;
}
--
```
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

● ● ●

From: keith@Apple.COM (Keith Rollin)
SubjectRe: SFGetFolder

In article <9741@zodiac.ADS.COM> jtn@ads.com (John Nelson) writes:
>In article <709@maytag.waterloo.edu> jb@aries5.UUCP () writes:
>>Has anybody written a SFGetFolder/SFPutFolder routine -- What I want to be
>> able to do is select or create a folder in a nice manner.
>>
>>The problem that I have with the implementations that I have seen of
>>selecting a folder i.e. 'Set Transfer Directory...'

MacDTS has released a sample program that shows you how to do this, among other tricks with StdFile. You can get this from APDA, BBS's and finer FTP sites everywhere, including ours. Basically, it involves writing a file filter procedure that always returns TRUE. Here is the relevant code:

```
FUNCTION FoldersOnly(p:ParmBlkPtr):BOOLEAN;

{ Normally, folders are ALWAYS shown, and aren't even passed to      }
{ this file filter for judgement. Under such circumstances, it is     }
{ only necessary to blindly return TRUE (allow no files whatsoever).  }
{ However, Standard File is not documented in such a manner, and      }
{ this feature may not be TRUE in the future. Therefore, we DO check  }
{ to see if the entry passed to us describes a file or a directory.   }

   BEGIN
       FoldersOnly := TRUE;
       IF BTst(p^.ioFlAttrib,4) THEN FoldersOnly := FALSE;
   END;
```

Hope this helps,

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

●●●

From: unknown
Subject: Re: HFS Help

This point is clearly covered in Inside Mac v. 4. the "vRef" of a folder is really a "wRef" a working directory reference. It only lives for a short time. You cannot depend on it from session to session. There is another number, a 4-byte dirId that is the permanent identifier of the folder, even if the folder is renamed. There is a system call that takes a wRef and a real volume reference and returns the dirId. PBHOpen() takes a volume Id, a dirId, and a file name and opens the file. To store a file descriptor from run to run, you need a triple: the volume name, the dirId, and the file name. To be really safe, you need the full path name too, so you can use it on a remote file system that doesn't really support dirIds. You need the volume name as a string since it may get a different number depending on the order of the mounting of the disks at boot time.

●●●

From: mm5l+@andrew.cmu.edu (Matthew Mashyna)
Subject: Re: INIT getting it's file name... (with Code)

Ando Sonenblick: sonenbli@oxy.edu writes:
        >...is there any way
        >for an INIT to get the name of the file it is in? I need to later open
        >the resource fork of the file and I want the user to be able to change the
        >name of the file; hence, my INIT needs to be able to read in the file name.

Check the latest issue of MacTutor: Writing INITs in Think C, by J.Peter Hoddie.

```
findMyName(name)
Str255 name;
{
        FCBPRec p;

        p.ioCompletion = 0;
        p.ioRefNum = CurResFile();
        p.ioVRefNum = 0;
        p.ioNamePtr = (StringPtr) name;

        PBGetFCBInfo(&p,false);
```

```
        BlockMove( p.ioNamePtr,&name, 1 + *(char *)(p.ioNamePtr) );
}
```

Matt Mashyna

●●●

From: svc@well.UUCP (Leonard Rosenthol)
Subject**Re: INIT getting it's file name...(with Code)**

In article <4020@helios.ee.lbl.gov> beard@ux1.lbl.gov (Patrick C Beard) writes:
> >In article <56505@tiger.oxy.edu> sonenbli@oxy.edu (Andrew D. Sonenblick) writes:
> >>
> >>   Ok, this is a quick one but it is very important: is there any way
> >>for an INIT to get the name of the file it is in? I need to later open
> >>the resource fork of the file and I want the user to be able to change the
> >>name of the file; hence, my INIT needs to be able to read in the file name.
> >>Please inform me or the whole net of any solution. Thanks.

I don't know what all this mess is about searching the System Folder, etc. as that is a REAL PAIN, and it also doesn't provide help for when you use some INIT Manager that supports subFolders...So here is the piece of code that I use for getting not only my INIT's name, but also the vRefNum (OK, WDRefNum) where I am.

```
void GetInitInfo(myName, myVRefNum)
Str255  *myName;
int     *myVRefNum;
{
        FCBPBRec        p;
        int                     dummy;

        p.ioCompletion = 0;
        p.ioRefNum = CurResFile();
        p.ioFCBIndx = 0;
        p.ioVRefNum = 0;
        p.ioNamePtr = (StringPtr)myName;
        dummy = PBGetFCBInfo(&p, FALSE);

        BlockMove(p.ioNamePtr, &myName, (long)p.ioNamePtr[0]);
        dummy = GetVol(NIL, myVRefNum);
}
```

[Pascal conversion is left as an exercise for the reader if required!]

--
Leonard Rosenthol | GEnie : MACgician

●●●

From: jackiw@cs.swarthmore.edu (Nick Jackiw)
Subject: Re: **opendir() and readdir()(enumerate dir code in pascal)**

h+@nada.kth.se (Jon W{tte) writes:
> I simply want to get the names of the files in adirectory.
> No, I am no novice to mac programming, but I've let libraries
> handle file I/O for me until now. I looked through IM IV, but
> the HFS chapter is less than crystal clear on this point...
>
> Now, how do I search {a | the current} directory for file
> names (and maybe types...) ? Any code pieces or hints are
> welcome !

File Manager

Try this.

```
program enumerateDir;
```

```
var

    ourParam: ParmBlkPtr;  {For finding ChainMail file in System Folder}
    ourWDParam: WDPBPtr;   {For opening a working directory}
    mailBoxAddr, mailFileName: str255;
    fileNum: integer;   {For indexing all SysFolder files}
    theErr: OSErr;      {Misc. OS Call error code}

begin
    MailBoxAddr:='Saturn:System Folder:'; {**whatever**}
    with ourWDParam^ do
        begin
            ioCompletion := nil;
            ioNamePtr := @MailBoxAddr;
            ioVRefNum := 0;
            ioWDProcID := 0;
            ioWDDirId := 0;
        end;

    if PBOpenWD(ourWDParam, false) <> noErr then
        begin
            writeln('Too many working directories open!');
            readln
        end;

    writeln('Here goes!');

    fileNum := 1;             {Begin examining all files}

    with ourParam^ do
        begin
            ioCompletion := nil;
            ioNamePtr := @mailFileName;
            ioVRefNum := ourWDParam^.ioVRefnum;
            mailFileName := '';
        end;

    repeat

        ourParam^.ioFDirIndex := fileNum;
        theErr := PBGetFInfo(ourParam, false);

        if theErr = noErr then
    begin
writeln(mailFileName, ' ', ourParam^.ioFlFndrInfo.fdType, ' ',
ourParam^.ioFlFndrInfo.fdCreator);
    end;

        fileNum := succ(fileNum);
    until theErr <> noErr;

end.
```

Sorry about all the tabs...vi seems to have a wider setting than LSP.

This is a fragment cut from working code. Reading it once, nothing looks spurious, but there may be references to the particular system in which it was formerly embedded. The last writln in the program should list the filename, file type, and creator field, as per your request.

Nicholas Jackiw [jackiw@cs.swarthmore.edu|jackiw@swarthmr.bitnet]

File Manager

●●●

From: tim@hoptoad.uucp (Tim Maroney)

Subject: Re: **Please help w/ PBSetCatInfo**

>In article <30340@shemp.CS.UCLA.EDU> tj@oahu.cs.ucla.edu (Tom Johnson) writes:
>> error=PBGetCatInfo(&myInfoRec,false);
>> if (error != noErr) SysBeep(10);
>> [flags set, etc.]
>> error=PBSetCatInfo(&myInfoRec,false);
>> if (error != noErr) Debugger();
>>
>>When I run this code, no error is detected on the PBSetCatInfo call and
>>the CInfoPBRec appears to be correctly filled in, but PBSetCatInfo
>>returns error=FFD5 a fnfError (file not found). Does anybody have
>>any idea why? This has been driving me crazy for hours!!

In article <37560@apple.Apple.COM> keith@Apple.COM (Keith Rollin) writes:
>I think that what is happening is that your ioDirID field is being changed from
>MyCurID on the PBGetCatInfo call. Then, when you try the PBSetCatInfo call,
>you are using a DirID different from the original one.

Yes. PBGetCatInfo changes the dirID it returns, apparently to the file ID. (Yet another place where this pointless feature is a pain in the ass.) It is necessary to save the dirID before calling PBGetCatInfo, then to reset the ioDirID field to this saved value before calling PBSetCatInfo. And of course, this is completely undocumented, unless you count the mysterious two-headed arrow before ioDirID in the description of the PBGetCatInfo routine.
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From: cc100aa@prism.gatech.EDU (Ray Spalding)
Subject: Re: **FSRead hangs on the serial driver**

In article <7770@unix.SRI.COM> gil@ginger.sri.com (Gil Porat) writes:
[regarding FSRead with the serial driver]
>1) Why does the second FSRead hang?

Because it will not return until it gets the exact number of characters you asked it to read.

>2) Doesn't FSRead return as much as it can?

No, see above.

>3) Is there a way to get the serial driver and/or FSRead to time out?

The way to accomplish this is to use asynchronous I/O (in the sense of IM vol IV, not in the sense of async data communications). You start an asynchronous read request, then go about your business processing events and so on. Periodically, you check the request block for completion: 0 == normal completion; 1 == still in progress; anything else == error. If you want to time it out, you have to watch the time yourself (using TickCount).

Caveat: Be sure to call SystemTask (or WaitNextEvent) periodically so that the serial driver will get a slice of CPU.

Some fragments of the code I use (MPW C):

```
ParamBlockRec commInPB;
static unsigned char commInChar;
static short commInPort = -6;

CommStartIn() /* Queue serial port input request and return */
{
    commInPB.ioParam.ioCompletion = NULL;
    commInPB.ioParam.ioVRefNum = 0;
    commInPB.ioParam.ioRefNum = commInPort;
```

```
commInPB.ioParam.ioBuffer = &commInChar;
commInPB.ioParam.ioReqCount = 1; /* ask for one character only */
commInPB.ioParam.ioPosMode = 0;
PBRead(&commInPB,true);
```

```
}

CommDisp() /* Check for serial port input completion */
{
    int rc;

    SystemTask();
    rc = commInPB.ioParam.ioResult;
    if (rc == 1) return; /* or check for timeout here */
    if (rc != noErr) { /* process errors */ return;}
    /* ... process one input character ... */
    CommStartIn(); /* queue request for next character */
    return;
}
```

>4) Does FSRead need an EOT to return?

No, EOT is passed through to you like any other character.

--
Ray Spalding

●●●

From: chesley@goofy.apple.com (Harry Chesley)
Subject: Re: FSRead hangs on the serial driver

In article <4713@hydra.gatech.EDU> cc100aa@prism.gatech.EDU (Ray Spalding) writes:

> >1) Why does the second FSRead hang?
>
> Because it will not return until it gets the exact number of characters
> you asked it to read.

More exactly: it will read until it gets at least as many characters as you ask for (more is OK).

It sounds like you're experiencing either buffer overflow or baud rate mismatch. The default input buffer for the serial port is only 64 bytes. So 80 characters can easily overflow the buffer if you don't read it out fast. You can increase the buffer size by calling SerSetBuf (but be sure and unset the buffer by calling SerSetBuf with a length of zero before exiting the application or you'll get horrible crashes).

Alternatively, the baud rates may not be set right. Therefore, you send enough characters at one speed, but they come through as fewer gibberish characters at the other speed.

In article <4713@hydra.gatech.EDU> cc100aa@prism.gatech.EDU (Ray Spalding) writes:
> >3) Is there a way to get the serial driver and/or FSRead to time out?
>
> The way to accomplish this is to use asynchronous I/O (in the
> sense of IM vol IV, not in the sense of async data communications).
> You start an asynchronous read request, then go about your business processing
> events and so on. Periodically, you check the request block for completion:
> 0 == normal completion; 1 == still in progress; anything else == error.
> If you want to time it out, you have to watch the time yourself (using
> TickCount).

Even simpler, use SerGetBuf to find out how many input characters are available, and do a synchronous call but only when there are enough characters available.

●●●

File Manager

In article <38534@cornell.UUCP>, wayner@svax.cs.cornell.edu (Peter Wayner) writes:

> I've been coding up an application which needs to keep a list of files
> and their locations on the disk. This list needs to be stable from boot
> to boot and even between restores from backup. There are several ways
> this can be done:
>
> 1) Full Path name
> 2) file name and DirId

2) should be volume name, file name, and DirID.

> My questions are;
>
> Are there any other problems with pathnames?

The other problem with pathnames is that they may exceed 255 bytes in length, which may make them difficult to use.

The best approach would be to store both (1) and (2), with code to handle the 256+ length problem mentioned above. When accessing the file, use (2); if that works, update (1). If (2) doesn't work, use (1) and update (2). However, few if any products actually do this; my own current products use (2) only. One of them (PowerStation) has a command to search all online disks for "lost" files, as typically occurs after an initialize/restore.

> How can I convert a DirId into a vRef which I can then use with [FSOpen]?
> Or more generally, how can I open a file with a name and a DirId?

As noted above, you will nead a volume specifier in addition to file name and DirID. Instead of using FSOpen, you can use PBHOpen. But to answer the question: you would create a WD with OpenWD, and then pass a WDRefNum instead of a VRefNum to FSOpen. Since WDs occupy a system-wide table of limited size, you would clean up afterwards with CloseWD.

> What about System 7.0? The tech note mentions that file id numbers will
> be the best when System 7.0 comes around because they will be stable
> even after renaming.

Right, but it ain't here yet; also, you will probably want your application to run under earlier systems. In a couple of years or so it will be feasible to require System 7 or later, just as circa 1988 it became feasible to require the Mac Plus feature set (introduced in early 1986).
--
brecher@well.sf.ca.us (Steve Brecher)

●●●

From: tim@hoptoad.uucp (Tim Maroney)
Subject: Re: SFReply.vRefNum -> PathName (pathname from vRefnum and Back in C)

The Pascal source posted ignores errors and string lengths, and so it can easily crash your machine if the path name runs to more than 255 characters or if an error leaves directoryName with a large length byte. Here's some C code to do the same thing (MPW C 3.0, but since it was originally written in LSC 3.0 it shouldn't be hard to convert back). FullFileName should be passed an empty string in tail if you just want the name of a folder.

```
SFVolDir(short wd, short *volume, long *folder)
{   WDPBRec pb; unsigned char name[72];
    pb.ioNamePtr = (StringPtr)name;
    pb.ioVRefNum = wd;
    pb.ioWDIndex = 0;
    pb.ioWDProcID = 0;
    pb.ioWDVRefNum = 0;
    PBGetWDInfo(&pb, false);
    if (pb.ioResult) return pb.ioResult;
```

File Manager

```
*volume = pb.ioWDVRefNum;
*folder = pb.ioWDDirID;
```

```
    return noErr;
}

FullFileName(StringPtr outname, StringPtr tail, short volume, long dirID)
{   StringHandle name; Str255 text, volname; HVolumeParam hvp;
    CInfoPBRec di; long size; short err;

    /* extract the volume name */
    hvp.ioNamePtr = volname;
    hvp.ioVRefNum = volume;
    hvp.ioVolIndex = 0;
    PBHGetVInfo((HParmBlkPtr)&hvp, false);
    if (hvp.ioVSigWord == 0xd2d7) { outname[0] = 0; return noHFSerr; }

    /* create and initialize the name handle */
    if (tail) {
        if (err = PtrToHand(tail + 1, &name, tail[0])) return err;
        size = tail[0];
    }
    else { if ((name = NewHandle(0)) == 0) return MemError();
        size = 0;
    }

    /* now start extracting the dirs and prepending them to
     * the handle
     */
    for ( ; dirID != 2; dirID = di.dirInfo.ioDrParID) {
        text[0] = 0;
        di.dirInfo.ioNamePtr = text;
        di.dirInfo.ioVRefNum = volume;
        di.dirInfo.ioFDirIndex = -1;
        di.dirInfo.ioDrDirID = dirID;
        PBGetCatInfo(&di, false);
        text[++text[0]] = ':';
        SetHandleSize(name, size += text[0]);
        BlockMove((Ptr)*name, (Ptr)(*name) + text[0], size - text[0]);
        BlockMove((Ptr)text + 1, (Ptr)*name, text[0]);
    }

    /* prepend the volume name onto the handle */
    volname[++volname[0]] = ':';
    SetHandleSize(name, size += volname[0]);
    BlockMove((Ptr)*name, (Ptr)(*name) + volname[0], size - volname[0]);
    BlockMove((Ptr)volname + 1, (Ptr)*name, volname[0]);

    /* copy and delete the handle */
    if (size > 255) {
        DisposHandle(name);
        SysBeep(12);
        outname[0] = 0;
        return tooBigErr;
    }
    outname[0] = size;
    BlockMove((Ptr)*name, (Ptr)outname + 1, size);
    DisposHandle(name);
    return noErr;
}
```

And on the theory that you might want to convert the name back some day, here's a routine I use for that. Notice that it will actually create the directory if it doesn't exist now. If this isn't what you want, throw away the error recovery code.

File Manager

```
FullToFolder(StringPtr name, short *volume, long *folder)
{   CInfoPBRec dirInfo; HVolumeParam hvp; short i;
```

```
    dirInfo.dirInfo.ioNamePtr = name;
    dirInfo.dirInfo.ioDrDirID = 0;
    dirInfo.dirInfo.ioVRefNum = 0;
    dirInfo.dirInfo.ioFDirIndex = 0;
    if (PBGetCatInfo (&dirInfo, 0) == noErr) {
        if (dirInfo.dirInfo.ioFlAttrib & 0x10) {
            *folder = dirInfo.dirInfo.ioDrDirID;
            hvp.ioNamePtr = name;
            for (i = 0; i < name[0]; i++)
                if (name[i+1] == ':')
                    { name[0] = i + 1; break; }
            hvp.ioVolIndex = -1;
            hvp.ioVRefNum = 0;
            PBHGetVInfo ((HParmBlkPtr)&hvp, 0);
            *volume = hvp.ioVRefNum;
        }
        else { *volume = 0, *folder = 0; return -1; }
    }
    else if (dirInfo.dirInfo.ioResult == fnfErr
         || dirInfo.dirInfo.ioResult == dirNFErr)
    {   dirInfo.dirInfo.ioNamePtr = (StringPtr)name;
        dirInfo.dirInfo.ioDrDirID = 0;
        dirInfo.dirInfo.ioVRefNum = 0;
        if (PBDirCreate((HParmBlkPtr)&dirInfo, false) == noErr) {
            *volume = dirInfo.dirInfo.ioVRefNum;
            *folder = dirInfo.dirInfo.ioDrDirID;
        }
        else { *volume = 0, *folder = 0;
            return dirInfo.dirInfo.ioResult;
        }
    }
    else  { *volume = 0, *folder = 0; return dirInfo.dirInfo.ioResult; }
    return noErr;
}
```
--
Tim Maroney, Mac Software Consultant, sun!hoptoad!tim, tim@toad.com

●●●

From:Bill Stackhouse
Subject:Create a folder code (in Pascal)

{The other day, someone posted a source for creating folders along a pathname.

I have updated that concept to include:

>   - specify a full path name
>   - use the volume name specifed in the path name
>   - use the file name spedifed in the path name
>   - if one or more directories in path already exist, use them
>   - if a specified directory has the same name as an existing file, stop
>   - allow specification of a volume name as a volref num
>   - allow specification of a starting directory as a dirID
>   - return the volref num and the dirID of the final directory created

Bill Stackhouse
bills@XAIT.Xerox.COM}


```
unit UCreateFolder;
```

File Manager

```
interface

function CreateFolder (pName: string;
     var pVRefNum: Integer;
```

```
      var pDirID: Integer): OSErr;

{CreateFolder - create all folders and files in a pathname. }
{ pName - vol:dir:dir: ... :file }
{         if vol ommited, will use pVRefNum }
{         if file ommited, only the directories will be created. }
{pVRefNum - input: if pName does not specify vol, the volRefNum. }
{         output: the ioVRefNum used to create the directories and file. }
{pDirID - input: if not 0, the dirID where to start creating directories. }
{         output: the ioDirID of the last directory in the list. }
{}
{ Result codes }
{   aabbb where aa = the vol/dir/file which caused the }
{       error and bbb = the error. }
{   example:   }
{       input a:b:c  or  :b:c   with result = -2048 is 'c' is dup file name. }
{       input a:b   with result = -0035 is 'a' not mounted. }
{ noErr   - ok }
{ nsvErr  - vol name not mounted }
{ dupFNErr  - either file exists in directory or needed to }
{         create folder with same name as existing file. }
{         Existing folders with same name are not errors. }

implementation

 function CreateFolder (pName: string;
      var pVRefNum: Integer;
      var pDirID: Integer): OSErr;

   var
     i: Integer;
     count: Integer;
     err: OSErr;
     theParms: HParamBlockRec;
     theCat: CInfoPBRec;
     theName: string[31];
 begin
   theParms.ioCompletion := nil;
   theParms.ioResult := 0;
   theParms.ioVRefNum := pVRefNum;
   theParms.ioDirID := pDirID;
   theCat.ioCompletion := nil;
   theCat.ioResult := 0;
   theCat.ioVRefNum := theParms.ioVRefNum;
   theCat.ioFDirIndex := 0;
   theCat.ioDrDirID := theParms.ioDirID;
   i := Pos(':', pName);
   theName := Copy(pName, 1, i - 1);
   Delete(pName, 1, i);
   if Length(theName) > 0 then {set ioVRefNum to correct vol}
     begin
       theName := Concat(theName, ':');
       theParms.ioNamePtr := @theName;
       theParms.ioVolIndex := -1;
       err := PBHGetVInfo(@theParms, FALSE);
       if err = nsvErr then
         begin
           CreateFolder := err;
           Exit(CreateFolder);
         end;
       theParms.ioDirID := pDirID;
     end;
   count := 0;
```

File Manager

```
repeat
  count := count + 1;
  i := Pos(':', pName);
```

```
        if i = 0 then
          begin
            theName := Copy(pName, 1, Length(pName));
            Delete(pName, 1, Length(pName));
            theParms.ioNamePtr := @theName;
            err := PBHCreate(@theParms, FALSE);
          end
        else
          begin
            theName := Copy(pName, 1, i - 1);
            Delete(pName, 1, i);
            theParms.ioNamePtr := @theName;
            err := PBDirCreate(@theParms, FALSE);
            if err = DupFNErr then
              begin
                theCat.ioVRefNum := theParms.ioVRefNum;
                theCat.ioNamePtr := @theName;
                theCat.ioDrDirID := 0;
                err := PBGetCatInfo(@theCat, FALSE);
                if not BitTst(@theCat.ioFlAttrib, 3) then {found file with same n}
                  err := DupFNErr;
                theParms.ioDirID := theCat.ioDrDirID;
              end;
          end;
    until (Length(pName) = 0) or (err <> noErr);
    pVRefNum := theParms.ioVRefNum;
    pDirID := theParms.ioDirID;
    if err <> noErr then
      err := err - (count * 1000);
    CreateFolder := err;
  end;    {CreateFolder}

end.
```

●●●

From: Apple DTS
Subject: How do I get a pathname from a working directory? (full Code)

```
(** PathNameFromDirID ***************************************************)
(*
(*  Given a DirID and real vRefnum, this routine will create and return the
(*  full pathname that corresponds to it. It does this by calling PBGetCatInfo
(*  for the given directory, and finding out its name and the DirID of its
(*  parent. It the performs the same operation on the parent, sticking ITS
(*  name onto the beginning of the first directory. This whole process is
(*  carried out until we have processed the root directory (identified with
(*  a DirID of 2.
(*
(*  NOTE: This routine is now A/UX friendly. A/UX likes sub-directories
(*        separated by slashes in a pathname. This routine automatically
(*        uses colons or slashes as separators based on the value of the
(*        global gHasAUX.  This global must be initialized correctly for
(*        this routine to do its thing. However, because of this dependancy
(*        on the idiosyncracies of file systems, generating full pathnames
(*        for other than display purposes is discouraged; it's changed in
(*        the past when A/UX was implemented, and it may change again in
(*        the future it support for other file systems such as ProDOS,
(*        MS-DOS, or OS/2 are added.
(*
(***************************************************************************)
```

File Manager

```
FUNCTION PathNameFromDirID (DirID:longint; vRefnum:integer):str255;

  VAR
```

```
        Block : CInfoPBRec;
        directoryName, FullPathName : str255;

    BEGIN
      FullPathName := '';
      WITH block DO BEGIN
        ioNamePtr := @directoryName;
        ioDrParID := DirId;
      END;

      REPEAT
        WITH block DO BEGIN
          ioVRefNum := vRefNum;
          ioFDirIndex := -1;
          ioDrDirID := block.ioDrParID;
        END;
        err := PBGetCatInfo(@Block,FALSE);

        IF haveAUX THEN BEGIN
          IF directoryName[1] <> '/' THEN BEGIN
            { If this isn't root (i.e. "/"), append a slash ('/') }
            directoryName := concat(directoryName, '/');
          END;
        END ELSE BEGIN
          directoryName := concat(directoryName,':');
        END;
        fullPathName := concat(directoryName,fullPathName);

      UNTIL (block.ioDrDirID = 2);

      PathNameFromDirID := fullPathName;
    END;


(** PathNameFromWD **********************************************)
(*
(*  Given an HFS working directory, this routine returns the full pathname
(*  that corresponds to it. It does this by calling PBGetWDInfo to get the
(*  VRefNum and DirID of the real directory. It then calls PathNameFromDirID,
(*  and returns its result.
(*
(****************************************************************************)

 FUNCTION PathNameFromWD(vRefNum:longint):str255;

    VAR
      myBlock : WDPBRec;

    BEGIN

      {
      { PBGetWDInfo has a bug under A/UX 1.1.  If vRefNum is a real vRefNum
      { and not a wdRefNum, then it returns garbage.  Since A/UX has only 1
      { volume (in the Macintosh sense) and only 1 root directory, this can
      { occur only when a file has been selected in the root directory (/).
      { So we look for this and hardcode the DirID and vRefNum. }

      IF (haveAUX) AND (vRefNum = -1) THEN BEGIN

        PathNameFromWD := PathNameFromDirID(2,-1);

      END ELSE BEGIN
```

```
WITH myBlock DO BEGIN
  ioNamePtr := NIL;
  ioVRefNum := vRefNum;
```

```
      ioWDIndex := 0;
      ioWDProcID := 0;
    END;

  { Change the Working Directory number in vRefnum into a real vRefnum }
  { and DirID. The real vRefnum is returned in ioVRefnum, and the real }
  { DirID is returned in ioWDDirID. }

    err := PBGetWDInfo(@myBlock,FALSE);

    WITH myBlock DO
      PathNameFromWD := PathNameFromDirID(ioWDDirID,ioWDVRefnum)
  END;
END;
```

●●●

From: lefty@twg.com (David N. Schlesinger)
Subject: Re: How to get full pathname?

In article <PETE.90May12102735@tone.rice.edu> pete@tone.rice.edu (Pete Keleher) writes:

> For an application that I have been working on, I'd like to be able display
> either just the filename, or the complete path name.

Assuming that you are starting out with a filename and a directory ID, the way to proceed is as follows:

1] Call GetCatInfo with ioFDirIndex = -1, and with your base directory ID in ioDrDirID. When the call returns, you'll have the parent directory ID in ioDrParID and the name of the initial directory in the buffer pointed to by ioNamePtr. Prepend the name to your file name with a colon in between.

2] Call GetCatInfo again, the same way, but put the parent directory's ID into ioDrDirID. Again, prepend the name returned to the string you're building, with a colon between the new piece and what you've already built. Repeat this step until you get back an ioDrParID of 2; this indicates the root. Do it one last time to get the name of the root directory. Prepend this (again followed by a colon) to the string you've built so far.

Voila! A full pathname! Be careful, though. Full pathname can easily exceed 255 characters. This can cause major problems if you're using Str255s instead of c strings...

Hope this helps...

David N. Schlesinger

●●●

From: tim@efi.com (Tim Maroney)
Subject: Re: "SetVol-ing" to a folder in the System Folder...

In article <101186@tiger.oxy.edu> sonenbli@oxy.edu (Andrew D. Sonenblick) writes:

> Anyway, to the business at hand: I have tried many combinations of
>code, many variations of paramBlocks, etc. and yet I still cannot set the
>volume to a specific folder (I am not using SFGetFile()).
> What I do want to do is an effective "SetVol" to a folder (the name of
>which I know) which is in the System Folder (the vRefNum of which I know) so
>that if I call "OpenResFile" for example the Mac would attempt to open a file
>which is inside the folder in question.

What you should do is take the SysEnvirons working directory reference number for the system folder, and pass it to PBGetCatInfo in the ioVRefNum field, with ioFDirIndex set to zero and ioNamePtr set to the folder name. The directory id of

the folder you want will be returned in ioDrDirID. Now, armed with a working directory reference number and a directory id, you have to make a new working directory for the folder. First call PBGetWDInfo on the system folder WD reference number, to get the volume reference number (ioWDVRefNum). Then pass this volume reference number and the directory id from PBGetCatInfo to PBOpenWD. Finally you have a working directory

reference number for the folder. You can pass this to SetVol or PBSetVol. Don't use PBHSetVol, even though it lets you leave out a step or two -- see the relevant tech note on why this call is usually dangerous.

See? It's a snap! Why, any technical summa cum laude from a good ivy league school could do it in no more than a few months! The beauty and elegance of the Macintosh file system have never been surpassed.

● ● ●

From: austing@Apple.COM (Glenn L. Austin)
Subject: Re: Data Fork Use

wolf@mel.cipl.uiowa writes:

>I tried once before to get information on using the data fork for applications.
>Has anyone had experience with this? Can someone suggest a good source for
>information?

You can simply open the data fork. No big deal, but you DO have to get the name and location of the Application (PBGetFCBInfo, IM IV, pg. 179). The refNum of the resource file is the reference number returned from OpenResFile or CurResFile.

--
Glenn L. Austin

● ● ●

From: austing@Apple.COM (Glenn L. Austin)
Subject: Re: Data Fork Of Currently Running APPL

cl29+@andrew.cmu.edu (Cameron Christopher Long) writes:

>I am working on a program that stores data in its data fork. Currently,
>I am opening the data fork using FOPEN since I know the name of the APPL.
>However, if someone renames the program, how will I know what to open?

Use PBGetFCBInfo (IM4, pg. 179) with the refCon of the application's resource fork as the file refCon. This gives you the name of the application, whether it is locked, where it is on disk, etc.

>All suggestions are most appreciated,
>Cameron Altenhof-Long

You're welcome.

--
Glenn L. Austin

● ● ●

From: rmh@apple.com (Rick Holzgrafe)
Subject: Re: Data Fork Use

In article <42652@apple.Apple.COM> austing@Apple.COM (Glenn L. Austin) writes:

> wolf@mel.cipl.uiowa writes:
>
> >I tried once before to get information on using the data fork for applications.
> >Has anyone had experience with this? Can someone suggest a good source for
> >information?
>

> You can simply open the data fork. No big deal, but you DO have to get the
> name and location of the Application (PBGetFCBInfo, IM IV, pg. 179).
> The refNum of the resource file is the reference number returned from
> OpenResFile or CurResFile.

Perhaps easier is to get the app's name by calling GetAppParms (IM II, pg. 58) and using that and the default volume refnum (zero, or call GetVol at startup and save it) to do the open.

You can write in the data fork if you like, but then your app won't run right on locked or read-only volumes such as CD-ROMs and some file servers. Reading is no problem.

(Tried to mail this notion, but it bounced. Sorry to trouble the net.)

Rick Holzgrafe | {sun,voder,nsc,mtxinu,dual}!apple!rmh

●●●