

NOTE: This Technical Note has been [retired](#). Please see the [Technical Notes](#) page for current documentation.

# Technical Note PT570

## MPW Q&As

### CONTENTS

[Downloadables](#)

This Technical Note contains a collection of archived Q&As relating to a specific topic--questions sent the Developer Support Center (DSC) along with answers from the DSC engineers. Current Q&A's can be found on the [Macintosh Technical Q&A's web site](#).

[Oct 01 1990]

## MacBrowse and building or editing 411 files

Date Written: 1/26/93

Last reviewed: 6/14/93

We'd like to add documentation to the Documentation part of MacBrowse for our application. Is this possible and if so how? I've noticed that MacBrowse reads 411 files. How can I create a 411 index file?

MacBrowse will allow you to create but not edit your 411 files. Build your own 411 file by parsing your source files then using the **SaveAs Text** option in MacBrowse. The dialog box will have an option to save as 411. This will create a 411 file with stubs; then fill in the stubs using MPW or any text editor. The next time you open MacBrowse it will build the index file for your 411 file. For more information on the tags used in 411 files see the `Get` command documented in the MPW Command Reference on the E.T.O. CD under MPW Documentation. *Note:* The Copyright Notice is needed at the top of the 411 file.

OBJECT MASTER also has support for creating and editing 411 files. This product is available from Acius.

## Choice of C or Pascal language for Macintosh development

Date Written: 5/28/90

Last reviewed: 5/28/90

Should I use C or Pascal for Macintosh development? Which language is better? Which development system should I use?

Asking MacDTS which language is better is like asking the Rolling Stones what their favorite song is: the answer depends on whom you ask and the mood they are in.

Use the language that you are most comfortable with. MacDTS engineers use both C and Pascal. Both languages have advantages and disadvantages. Many people complain that C is terse and unreadable. Others say that Pascal is too verbose, and that its lack of knowledge about the machine can make programming awkward.

There are many good development systems. MPW is powerful and extensible, but it requires a lot of hardware to run efficiently. Both Think(TM) languages (from Symantec) are fast and easy to use, but they aren't extensible at all. There are also good languages from other vendors, including Semper Software(TM) Modula-2, Language Systems(TM) Fortran, MicroSoft(TM) and Zedcor(TM) BASIC, and Allegro Common Lisp from Apple (formerly from Coral Software).

Officially, Apple doesn't recommend any particular language or development system.

## Installing MPW 3.0

Date Written: 1/1/90

Last reviewed: 6/14/93

The MPW installer is reporting that it can't find the "Dolt" script. What is wrong and how can I properly install MPW?

The problems installing MPW 3.0 could be due to a limitation of the installation script--that is, using the Installer we provide will not work if there are single or double quotes, or other special characters, in the full pathname of the destination folder. This is noted in the MPW manual under "Getting Started." The solution is to either install MPW by hand (not hard unless you've never done it before and don't know where everything goes), or to rename your hard disk with a simpler name. Since most of MPW complaints occur when your hard disk's name has an apostrophe in it, I would suggest the latter method.

Another possible problem is caused by a bug in some third-party 'INIT's that cause the MPW installer to fail. This bug is caused by those 'INIT's using `PBHSetVol` and changing the default volume behind MPW's back. This problem is frequently reported by users with TOPS (from TOPS, a Sun Microsystems company).

Some 'INIT's use `PBHGetVol` to get and save the working directory, then attempt to restore it using `PBHSetVol`, but end up setting the working directory to the root instead. The Macintosh Technical Note "Why PBHSetVol is Dangerous" warns against using `PBHSetVol`.

Not only do these 'INIT's interfere with the installer they can also interfere with other MPW operations, most commonly when MPW tries to access a script in your current directory or a tool tries to open a script in the current directory.

To work around the installer problem, you can:

1. Manually install MPW;
2. Disable any 'INIT's (remove them from the System Folder and reboot), then run the installer. (Note: you can prevent TOPS from loading by holding down the Command key while you reboot);
3. Use a text editor to edit the file "Startup" in the Installation Folder. Search for the line starting with "Dolt" and insert the line "directory" immediately before it. This will force the MPW shell to reset the working directory. Save the Startup file and perform the installation procedure as documented.

Any 'INIT' that uses `PBHSetVol` will interfere with the setting of the working directory whenever a disk is inserted while MPW is running. You will have to issue a "directory" command to make the MPW shell reset the working directory.

## Directory can't be prerequisiteFile in an MPW dependency rule

Date Written: 1/28/91

Last reviewed: 6/14/93

We are working on a project with MPW. Is there a way to set up our makefile for the main project so that it is dependent on the source files from the library (already constructed), not the object file produced by running lib? Is there a way to refer to a directory (rather than a file) as a prerequisiteFile in a dependency rule?

That's a great suggestion, but unfortunately it isn't possible (at this time) to have a directory as a dependency. I will definitely forward the suggestion along to the appropriate people.

## Window-splitting resource format for MPW 3.2

Date Written: 7/29/91

Last reviewed: 6/14/93

The Macintosh window-splitting resource 'MPSR' (ID =1008) for MPW 3.2 contains a 20-byte header, followed by an array of 10-byte elements that describe how to make the panes. Its fields are defined as follows, subject to revision in future MPW releases:

The Header:

```
rect (8 bytes) Open Rectangle (zoomed out size)
rect (8 bytes) Close Rectangle (zoomed in size)
long (4 bytes) First position (offset into file of upper left corner)
```

Each Element:

```
enum (1 byte) Orientation - (none = 0, row = 1, column = 2)
pad (1 byte)
short (2 bytes) PaneSpec - an ID number, the pane to split
short (2 bytes) Size - offset from top or left for the split
```

The last element in the array is a null element with `Orientation = none`. The `PaneSpec` is used to locate the pane to split. A Pane spec consists of 5 3-bit fields. [0 eee ddd ccc bbb aaa]. Panes can be nested up to 5 levels deep. A pane can be split up to 7 times (3 bits).

## Script for finding Macintosh modify-read-only files

Date Written: 3/17/92

Last reviewed: 6/14/93

How can I tell if a given file (with projector ckid) has been `ModifyReadOnly'd`? What I really need to do is search a folder nest for all such files.

This is a script to find out where modify-read-only (or modifiable) files are. The files that are modify-read-only have a "\*" after the revision number; the modifiable files have a "+" after the revision number.

```
for file in `files -r -f`
projectinfo "{file}" -s
```

## MPW 3.0 max global data size is >32K, max local data still 32K

Date Written: 1/1/90

Last reviewed: 12/17/90

Is the maximum size for global and local data still 32K?

Starting with MPW 3.0, the maximum size for global data in MPW became larger than 32K, using compiler and linker options `-m` and `-srt`. Local data is tougher, because local data is allocated by using the `LINK` instruction--for example, `LINK A6,#$-380`. With this relative addressing mode, you're constrained to the negative side of a 16-bit value for local space, which translates to 32K. In other words, this limit is basically due to the Motorola processor architecture.

If you are allocating more than 32K either globally or locally, you might want to rearchitect your system to use dynamic (and theoretically unbounded, especially on virtual architectures) storage space.

## Calling/linking external routines

Date Written: 4/25/89

Last reviewed: 6/14/93

I'm trying to call external routines, written in assembler or some high-level language, from my C program, but the linker complains that my procedure is not defined anywhere. How can I make it accessible from C?

C is case sensitive, while some other languages, notably Pascal, are not. Pascal generates an all uppercase symbol names for any of your procedures. (For example, `MyProc` causes the Pascal compiler to create an entry point symbol named `MYPROC`.)

In C, if you are declaring a C routine, the name is case sensitive. For example:

requires an entry point named `MyProc`, requires an entry point named `MYPROC`, consistent with the Pascal compiler.

The assembler can be either case sensitive or case insensitive. Use the directive to make the assembler case sensitive like C. Make sure your assembler routines match their expected usage.

If all else fails, you can use the linker `-ma` option to remap entry point names to satisfy unresolved references.

[Back to top](#)

## Downloadables



Acrobat version of this Note (K).

[Download](#)

---

Technical Notes by [Date](#) | [Number](#) | [Technology](#) | [Title](#)  
[Developer Documentation](#) | [Technical Q&As](#) | [Development Kits](#) | [Sample Code](#)