NAME

perlmacosx - Perl under Mac OS X

SYNOPSIS

This document briefly describes Perl under Mac OS X.

```
 curl -O http://www.cpan.org/src/perl-5.24.1.tar.gz
tar -xzf perl-5.24.1.tar.gz
cd perl-5.24.1
./Configure -des -Dprefix=/usr/local/
make
make test
sudo make install
```

DESCRIPTION

The latest Perl release (5.24.1 as of this writing) builds without changes under all versions of Mac OS X from 10.3 "Panther" onwards.

In order to build your own version of Perl you will need 'make', which is part of Apple's developer tools - also known as Xcode. From Mac OS X 10.7 "Lion" onwards, it can be downloaded separately as the 'Command Line Tools' bundle directly from <https://developer.apple.com/downloads/> (you will need a free account to log in), or as a part of the Xcode suite, freely available at the App Store. Xcode is a pretty big app, so unless you already have it or really want it, you are advised to get the 'Command Line Tools' bundle separately from the link above. If you want to do it from within Xcode, go to Xcode -> Preferences -> Downloads and select the 'Command Line Tools' option.

Between Mac OS X 10.3 "Panther" and 10.6 "Snow Leopard", the 'Command Line Tools' bundle was called 'unix tools', and was usually supplied with Mac OS install DVDs.

Earlier Mac OS X releases (10.2 "Jaguar" and older) did not include a completely thread-safe libc, so threading is not fully supported. Also, earlier releases included a buggy libdb, so some of the DB_File tests are known to fail on those releases.

Installation Prefix

The default installation location for this release uses the traditional UNIX directory layout under /usr/local. This is the recommended location for most users, and will leave the Apple-supplied Perl and its modules undisturbed.

Using an installation prefix of '/usr' will result in a directory layout that mirrors that of Apple's default Perl, with core modules stored in '/System/Library/Perl/${version}', CPAN modules stored in '/Library/Perl/${version}', and the addition of '/Network/Library/Perl/${version}' to `@INC` for modules that are stored on a file server and used by many Macs.

SDK support

First, export the path to the SDK into the build environment:

```
export SDK=/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/
```

Please make sure the SDK version (i.e. the numbers right before '.sdk') matches your system's (in this case, Mac OS X 10.8 "Mountain Lion"), as it is possible to have more than one SDK installed. Also make sure the path exists in your system, and if it doesn't please make sure the SDK is properly installed, as it should come with the 'Command Line Tools' bundle mentioned above. Finally, if you have an older Mac OS X (10.6 "Snow Leopard" and below) running Xcode 4.2 or lower, the SDK path might be something like `'/Developer/SDKs/MacOSX10.3.9.sdk'`.

You can use the SDK by exporting some additions to Perl's 'ccflags' and '..flags' config variables:

```
./Configure -Accflags="-nostdinc -B$SDK/usr/include/gcc \
-B$SDK/usr/lib/gcc -isystem$SDK/usr/include \
-F$SDK/System/Library/Frameworks" \
-Aldflags="-Wl,-syslibroot,$SDK" \
-de
```

Universal Binary support

Note: From Mac OS X 10.6 "Snow Leopard" onwards, Apple only supports Intel-based hardware. This means you can safely skip this section unless you have an older Apple computer running on ppc or wish to create a perl binary with backwards compatibility.

You can compile perl as a universal binary (built for both ppc and intel). In Mac OS X 10.4 "Tiger", you must export the 'u' variant of the SDK:

```
export SDK=/Developer/SDKs/MacOSX10.4u.sdk
```

Mac OS X 10.5 "Leopard" and above do not require the 'u' variant.

In addition to the compiler flags used to select the SDK, also add the flags for creating a universal binary:

```
./Configure -Accflags="-arch i686 -arch ppc -nostdinc \
-B$SDK/usr/include/gcc \
-B$SDK/usr/lib/gcc -isystem$SDK/usr/include \
-F$SDK/System/Library/Frameworks" \
-Aldflags="-arch i686 -arch ppc -Wl,-syslibroot,$SDK" \
-de
```

Keep in mind that these compiler and linker settings will also be used when building CPAN modules. For XS modules to be compiled as a universal binary, any libraries it links to must also be universal binaries. The system libraries that Apple includes with the 10.4u SDK are all universal, but user-installed libraries may need to be re-installed as universal binaries.

64−bit PPC support

Follow the instructions in *INSTALL* to build perl with support for 64-bit integers (`use64bitint`) or both 64-bit integers and 64-bit addressing (`use64bitall`). In the latter case, the resulting binary will run only on G5-based hosts.

Support for 64-bit addressing is experimental: some aspects of Perl may be omitted or buggy. Note the messages output by *Configure* for further information. Please use `perlbug(1)` to submit a problem report in the event that you encounter difficulties.

When building 64-bit modules, it is your responsibility to ensure that linked external libraries and frameworks provide 64-bit support: if they do not, module building may appear to succeed, but attempts to use the module will result in run-time dynamic linking errors, and subsequent test failures. You can use `file` to discover the architectures supported by a library:

```
$ file libgdbm.3.0.0.dylib
libgdbm.3.0.0.dylib: Mach-O fat file with 2 architectures
libgdbm.3.0.0.dylib (for architecture ppc): Mach-O dynamically linked shared lib
libgdbm.3.0.0.dylib (for architecture ppc64): Mach-O 64-bit dynamically linked s
```

Note that this issue precludes the building of many Macintosh-specific CPAN modules (`Mac::*`), as the required Apple frameworks do not provide PPC64 support. Similarly, downloads from Fink or Darwinports are unlikely to provide 64-bit support; the libraries must be rebuilt from source with the appropriate compiler and linker flags. For further information, see Apple's *64-Bit Transition Guide* at <http://developer.apple.com/documentation/Darwin/Conceptual/64bitPorting/index.html>.

libperl and Prebinding

Mac OS X ships with a dynamically-loaded libperl, but the default for this release is to compile a static libperl. The reason for this is pre-binding. Dynamic libraries can be pre-bound to a specific address in memory in order to decrease load time. To do this, one needs to be aware of the location and size of all previously-loaded libraries. Apple collects this information as part of their overall OS build process, and thus has easy access to it when building Perl, but ordinary users would need to go to a great deal of effort to obtain the information needed for pre-binding.

You can override the default and build a shared libperl if you wish (Configure...-Duseshrplib).

With Mac OS X 10.4 ‘‘Tiger’’ and newer, there is almost no performance penalty for non-prebound libraries. Earlier releases will suffer a greater load time than either the static library, or Apple’s pre-bound dynamic library.

Updating Apple’s Perl

In a word - don’t, at least not without a *very* good reason. Your scripts can just as easily begin with ‘‘#!/usr/local/bin/perl’’ as with ‘‘#!/usr/bin/perl’’. Scripts supplied by Apple and other third parties as part of installation packages and such have generally only been tested with the /usr/bin/perl that’s installed by Apple.

If you find that you do need to update the system Perl, one issue worth keeping in mind is the question of static vs. dynamic libraries. If you upgrade using the default static libperl, you will find that the dynamic libperl supplied by Apple will not be deleted. If both libraries are present when an application that links against libperl is built, ld will link against the dynamic library by default. So, if you need to replace Apple’s dynamic libperl with a static libperl, you need to be sure to delete the older dynamic library after you’ve installed the update.

Known problems

If you have installed extra libraries such as GDBM through Fink (in other words, you have libraries under */sw/lib*), or libdlcompat to */usr/local/lib*, you may need to be extra careful when running Configure to not to confuse Configure and Perl about which libraries to use. Being confused will show up for example as ‘‘dyld’’ errors about symbol problems, for example during ‘‘make test’’. The safest bet is to run Configure as

```
Configure ... -Uloclibpth -Dlibpth=/usr/lib
```

to make Configure look only into the system libraries. If you have some extra library directories that you really want to use (such as newer Berkeley DB libraries in pre-Panther systems), add those to the libpth:

```
Configure ... -Uloclibpth -Dlibpth='/usr/lib /opt/lib'
```

The default of building Perl statically may cause problems with complex applications like Tk: in that case consider building shared Perl

```
Configure ... -Duseshrplib
```

but remember that there’s a startup cost to pay in that case (see above ‘‘libperl and Prebinding’’).

Starting with Tiger (Mac OS X 10.4), Apple shipped broken locale files for the eu_ES locale (Basque-Spain). In previous releases of Perl, this resulted in failures in the *lib/locale* test. These failures have been suppressed in the current release of Perl by making the test ignore the broken locale. If you need to use the eu_ES locale, you should contact Apple support.

Cocoa

There are two ways to use Cocoa from Perl. Apple’s PerlObjCBridge module, included with Mac OS X, can be used by standalone scripts to access Foundation (i.e. non-GUI) classes and objects.

An alternative is CamelBones, a framework that allows access to both Foundation and AppKit classes and objects, so that full GUI applications can be built in Perl. CamelBones can be found on SourceForge, at <http://www.sourceforge.net/projects/camelbones/>.

Starting From Scratch

Unfortunately it is not that difficult somehow manage to break one’s Mac OS X Perl rather severely. If all else fails and you want to really, **REALLY**, start from scratch and remove even your Apple Perl installation (which has become corrupted somehow), the following instructions should do it. **Please think twice before following these instructions: they are much like conducting brain surgery to yourself. Without anesthesia.** We will **not** come to fix your system if you do this.

First, get rid of the libperl.dylib:

```
# cd /System/Library/Perl/darwin/CORE
# rm libperl.dylib
```

Then delete every .bundle file found anywhere in the folders:

```
/System/Library/Perl
/Library/Perl
```

You can find them for example by

```
# find /System/Library/Perl /Library/Perl -name '*.bundle' -print
```

After this you can either copy Perl from your operating system media (you will need at least the /System/Library/Perl and /usr/bin/perl), or rebuild Perl from the source code with `Configure -Dprefix=/usr -Duseshrplib` NOTE: the `-Dprefix=/usr` to replace the system Perl works much better with Perl 5.8.1 and later, in Perl 5.8.0 the settings were not quite right.

''Pacifist'' from CharlesSoft (<http://www.charlessoft.com/>) is a nice way to extract the Perl binaries from the OS media, without having to reinstall the entire OS.

AUTHOR

This README was written by Sherm Pendley <sherm@dot-app.org>, and subsequently updated by Dominic Dunlop <domo@computer.org> and Breno G. de Oliveira <garu@cpan.org>. The ''Starting From Scratch'' recipe was contributed by John Montbriand <montbriand@apple.com>.

DATE

Last modified 2013-04-29.