OSX 10.5 Leopard Symbol Mangling with $non_lazy_ptr

Asked 16 years, 3 months ago Modified 13 years, 2 months ago

Viewed 6k times

3

Why does Leopard mangle some symbols with $non_lazy_ptr? More importantly what is the best method to fix undefined symbol errors because a symbol has been mangled with $non_lazy_ptr?

macos osx-leopard name-mangling darwin non-lazy-ptr

Share

Improve this question

Follow

edited Nov 4, 2008 at 2:45

S.Lott

391k ● 82 ● 517 ● 788

asked Sep 17, 2008 at 1:46

Erik

5 Answers

Sorted by: Highest score (default)

5

From: Developer Connection - Indirect Addressing

Indirect addressing is the name of the code generation technique that allows symbols defined in one file to be

referenced from another file, without requiring the referencing file to have explicit knowledge of the layout of the file that defines the symbol. Therefore, the defining file can be modified independently of the referencing file. Indirect addressing minimizes the number of locations that must be modified by the dynamic linker, which facilitates code sharing and improves performance.

When a file uses data that is defined in another file, it creates symbol references. A symbol reference identifies the file from which a symbol is imported and the referenced symbol. There are two types of symbol references: nonlazy and lazy.

Nonlazy symbol references are resolved (bound to their definitions) by the dynamic linker when a module is loaded. A nonlazy symbol reference is essentially a symbol pointer—a pointer-sized piece of data. The compiler generates nonlazy symbol references for data symbols or function addresses.

Lazy symbol references are resolved by the dynamic linker the first time they are used (not at load time). Subsequent calls to the referenced symbol jump directly to the symbol's definition. Lazy symbol references are made up of a symbol pointer and a symbol stub, a small amount of code that directly dereferences and jumps through the symbol pointer. The compiler generates lazy symbol references when it encounters a call to a function defined in another file.

Share Improve this answer

Follow

edited Oct 11, 2011 at 23:36

Colen

13.8k • 21 • 80 • 107

answered Sep 17, 2008 at 4:23

Brian Mitchell

2,288 • 14 • 12

3

In human-speak: the compiler generates stubs with $non_lazy_ptr appended to them to speed up linking. You're probably seeing that function Foo referenced from _Foo$non_lazy_ptr is undefined, or something like that - these are not the same thing. Make sure that the symbol is actually declared and exported in the object files/libraries you're linking your app to. At least that was my problem, I also thought it's a weird linker thing until I found that my problem was elsewhere - there are several other possible causes found on Google.

Share Improve this answer

Follow

answered Nov 14, 2008 at 14:19

Vladimir Panteleev

25.1k • 6 • 78 • 119

2

```
ranlib -c libwhatever.a
```

is a solid fix for the issue. I had the same problem when building the PJSIP library for iOS. This library sort-of uses an autoconf based make system, but needs a little tweaking to various files to make everything alright for iOS. In the process of doing that I managed to remove

the ranlib line in the rule for libraries and then started getting an error in the link of my project about `_PJ_NO_MEMORY_EXCEPTION` referenced from `_PJ_NO_MEMORY_EXCEPTION$non_lazy_ptr` being undefined.

Adding the ranlib line back to the library file solved it. Now my full entry for LIBS in rules.mak is

```
$(LIB): $(OBJDIRS) $(OBJS) $($(APP)_EXTRA_DEP)
    if test ! -d $(LIBDIR); then $(subst
@@,$(subst
/,$(HOST_PSEP),$(LIBDIR)),$(HOST_MKDIR)); fi
    $(LIBTOOL) -o $(LIB) $(OBJS)
    $(RANLIB) -c $(LIB)
```

Hope this helps others as well trying to use general UNIX configured external libraries with iPhone or iOS.

Share Improve this answer

Follow

answered Aug 11, 2010 at 20:59

Tom S.

179 • 3

1

If someone else stumbles the same problem I had:

Had a `extern NSString* const someString;` in the header file, but forgot to put it the implementation file. as `NSString* const someString=@"someString";`

This solved it.

Share Improve this answer

Follow

answered Jun 17, 2010 at 14:05

Ciryon
2,777 ● 3 ● 31 ● 35

ranlib -c on your library file fixes the problem

0

Share Improve this answer

Follow

answered Jun 28, 2009 at 20:34

Stewart Clark