dladdr doesn't return the function name

Asked 11 years, 2 months ago Modified 5 months ago Viewed 15k times

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email

G Sign up with Google

Sign up with GitHub

Sign up with Facebook

X

10/15/23, 9:30 AM 1 of 6



I'm trying to use dladdr. It correctly locates the library, but it does not find the function name. I can call objdump, do a little math, and get the address of the function that I pass dladdr. If objdump can see it, why can't dladdr?

Here is my function:

```
const char *FuncName(const void *pFunc)
{
Dl_info DlInfo;
int nRet;
    // Lookup the name of the function given the function pointer
    if ((nRet = dladdr(pFunc, &DlInfo)) != 0)
        return DlInfo.dli_sname;
    return NULL;
}
```

Here is a gdb transcript showing what I get.

```
Program received signal SIGINT, Interrupt.
[Switching to Thread 0xf7f4c6c0 (LWP 28365)]
0xffffe410 in __kernel_vsyscall ()
(gdb) p MatchRec8Cmp
2 = \{\text{void} (\text{TCmp}^*, \text{TWork}^*, \text{TThread}^*)\} 0 \times \text{f1b62e73} < \text{MatchRec8Cmp} > \text{MatchRec8Cmp}
(gdb) call FuncName(MatchRec8Cmp)
$3 = 0x0
(gdb) call FuncName(0xf1b62e73)
$4 = 0 \times 0
(gdb) b FuncName
Breakpoint 1 at 0xf44bdddb: file threads.c, line 3420.
(gdb) call FuncName(MatchRec8Cmp)
Breakpoint 1, FuncName (pFunc=0xf1b62e73) at threads.c:3420
3420
The program being debugged stopped while in a function called from GDB.
When the function (FuncName) is done executing, GDB will silently
stop (instead of continuing to evaluate the expression containing
the function call).
(gdb) s
                  if ((nRet = dladdr(pFunc, &DlInfo)) != 0)
3426
(gdb)
3427
                           return DlInfo.dli_sname;
(gdb) p DlInfo
$5 = {dli_fname = 0x8302e08 "/xxx/libdata.so", dli_fbase = 0xf1a43000, dli_sname
= 0 \times 0, dli_saddr = 0 \times 0}
(gdb) p nRet
$6 = 1
(gdb) p MatchRec8Cmp - 0xf1a43000
$7 = (\text{void } (*)(\text{TCmp } *, \text{TWork } *, \text{TThread } *)) 0x11fe73
(gdb) q
The pregram is rupping Evit anyway? (y or n) y
```

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email

Sign up with Google

Sign up with GitHub

Sign up with Facebook

X

```
$ objdump --syms /xxx/libdata.so | grep MatchRec8Cmp
0011fe73 l F .text 00000a98 MatchRec8Cmp
```

Sure enough, 0011fe73 = MatchRec8Cmp - 0xf1a43000. Anyone know why dladdr can't return dli sname = "MatchRec8Cmp" ???

I'm running Red Hat Enterprise Linux Server release 5.4 (Tikanga). I have seen this work before. Maybe it's my compile switches:

```
CFLAGS = -m32 -march=i686 -msse3 -ggdb3 -pipe -fno-common -fomit-frame-pointer \
    -Ispio -fms-extensions -Wmissing-declarations -Wstrict-prototypes
-Wunused -Wall \
    -Wno-multichar -Wdisabled-optimization -Wmissing-prototypes -Wnested-
externs \
    -Wpointer-arith -Wextra -Wno-sign-compare -Wno-sequence-point \
    -I../../include -I/usr/local/include -fPIC \
    -D$(Uname) -D_REENTRANT -D_GNU_SOURCE
```

I have tried it with -g instead of -ggdb3 although I don't think debugging symbols have anything to do with elf.

c linux elf

Sign up with email

Share Improve this question Follow



asked Jul 30, 2012 at 23:41 johnnycrash 5,214 5 34 58

Sign up with Facebook

X

Just a guess - try to extern "C" your MatchRec8Cmp()? – YePhlcK Jul 30, 2012 at 23:55

Worth a try, cept I don't think the names looked mangled when I did objdump and the funcs are in .c files. – johnnycrash Jul 31, 2012 at 0:16

Did you pass -rdynamic at linking time of your executable? – Basile Starynkevitch Jul 31, 2012 at 3:36

@BasileStarynkevitch -rdynamic is unlikely to help: it's the default when linking a shared library anyway. – Employed Russian Jul 31, 2012 at 4:05

I was talking of -rdynamic for the program executable (and it is not the default in that case) - not for shared libraries. – Basile Starynkevitch Jul 31, 2012 at 10:31



Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with GitHub

Sign up with Google

3 of 6 10/15/23, 9:30 AM

If objdump can see it, why can't dladdr

11

dladdr can only see functions exported in the dynamic symbol table. Most likely



nm -D /xxx/libdata.so | grep MatchRec8Cmp



shows nothing. Indeed your objdump shows that the symbol is *local*, which proves that this is the cause.



The symbol is local either because it has a hidden visibility, is static, or because you hide it in some other way (e.g. with a linker script).

Update:

Those marked with the 'U' work with dladdr. They get "exported" automatically somehow.

They work because they are exported from *some other shared library*. The $\,\upsilon\,$ stands for unresolved, i.e. defined elsewhere.

Update 2023/05/14:

I see that there are a few "answers" below which tell you to add <code>-rdynamic</code> or <code>--export-dynamic</code> to "solve" the problem.

These answers don't explain the "why" (i.e. they aren't an actual answer to the question that was asked), and also don't explain the cost of the solution, which could be significant.

First, adding -rdynamic flag is preferable, because that is a the compiler front-end flag, which gets translated into *appropriate* linker flag (some linkers understand -E, some understand --export-dynamic, some understand both).

Second, if you are going to add a linker-specific flag, you should do it *correctly*: -wl, -- export-dynamic . Adding --export-dynamic without -wl, prefix *happens to work* by accident -- GCC doesn't understand that flag and passes it to the linker. But it may do something else in the future.

What are the easts of _____ 2 It clove down your evecutable leading. How much of a

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email



Sign up with GitHub

Sign up with Facebook

X

4 of 6 10/15/23, 9:30 AM

symbol in a shared library needs to be resolved.

This can also break your executable if the symbol you *didn't want* to be exported becomes exported.

Are there better solutions than -rdynamic?

Glad you asked. There probably are.

Newer versions of the linker have --export-dynamic-symbol SYMBOL and --export-dynamic-symbol-list FILE options (do use -wl, prefix if you are going to pass these flags to GCC), which allow you to control *exactly* which symbols are exported.

That is a much more precise solution (compared to -rdynamic which exports *everything*), and the cost is (usually) significantly lower.

Share Improve this answer

Follow

edited May 14 at 19:16

answered Jul 31, 2012 at 4:00



Employed Russian

1k 34 297

You are correct in that the function in question does not show up when I do nm -D. In fact a small random sampling of my functions is returned when I call nm. When I pass the address of a function that

does show up, dladdr works. What I can't figure out is how to export the function. I don't use -fvisibility, so by default all functions should be exported. I tried using **attribute** ((visibility("default"))), but that doesn't work either. — johnnycrash Jul 31, 2012 at 14:45

I hunted around for a long time and found a bash script being executed in the build which set a make variable which resulted in a --version-script being added to the build. However for some reason, there are a number of functions that show up in the nm -D result that are not listed in the --version-script. They all look like this: "U MemInfoTransCreate", whereas the items in the --version-script that are marked as "global", look like this: "0000000001202ff T MatchRec8Cmp_th". Those marked with the 'U' work with dladdr. They get "exported" automatically somehow. — johnnycrash Jul 31, 2012 at 18:06

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

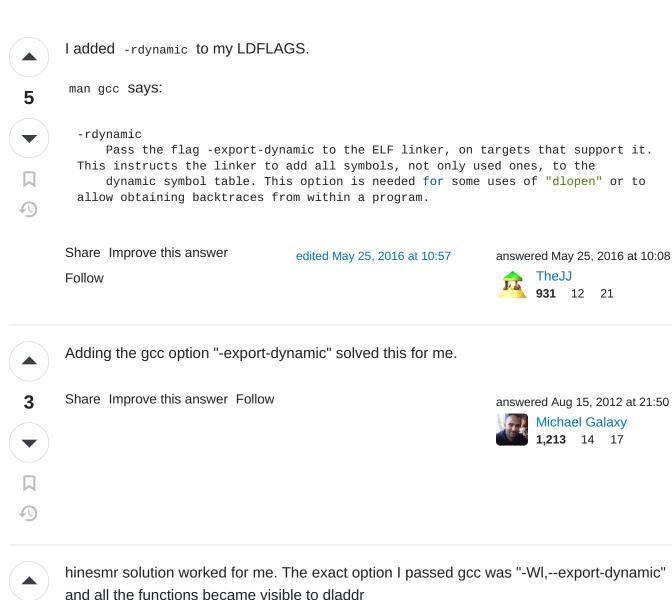
Sign up with email

G Sign up with Google

Sign up with GitHub

Sign up with Facebook

X



and all the functions became visible to diaddr

Share Improve this answer Follow

answered Feb 9, 2013 at 3:48

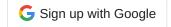
Michael Gruner

572 4 7

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email

0



Sign up with GitHub

Sign up with Facebook

X