Support for mini-debuginfo in LLDB

How to read the .gnu_debugdata section

Konrad Kleine

February 2, 2020

Red Hat

🐞 About me

👤 Konrad Kleine
- Red Hat
- LLDB, C/C++, ELF, DWARF since 2019
- Before worked OpenShift since 2016

💬 Reach out
- 🐙 https://github.com/kwk/talks/
- 💼 https://www.linkedin.com/in/konradkleine
- 📡 https://developers.redhat.com/blog/author/kkleine/
- 🐦 https://twitter.com/realdonaldtrump

2

✤ Overall goal and first steps

Improve LLDB for Fedora and RHEL binaries
when no debug symbols installed

Take existing Fedora binary (/usr/bin/zip)

- identify a symbol/function
- shootout: GDB vs. LLDB
- hurdles:
    - not from .dynsym
    - from within .gnu_debugdata

What is the `.gnu_debugdata` section (aka mini-debuginfo)?
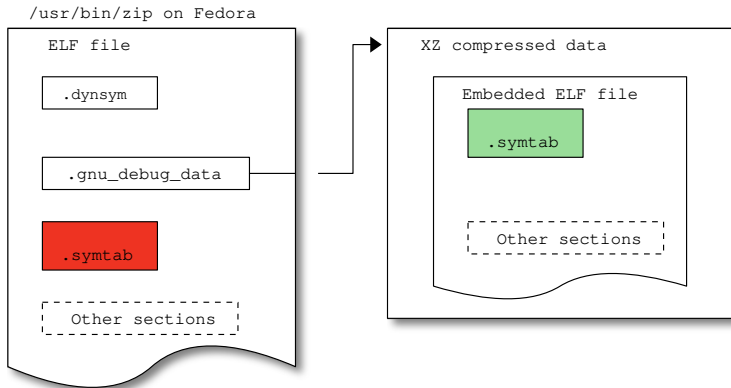
- **extra minimal** debug info for simple backtraces
- **no replacement** for separate full debug info
- **not related** to DWZ compression

🗁 **Extract and uncompress** `.gnu_debugdata` **section to** `zip.gdd`

```
1  ~$ cp /usr/bin/zip .
2  ~$ objcopy --dump-section .gnu_debugdata=zip.gdd.xz zip
3  ~$ file zip.gdd.xz
4  zip.gdd.xz: XZ compressed data
5  ~$ xz --decompress --keep zip.gdd.xz
6  ~$ file zip.gdd
7  zip.gdd: ELF 64-bit LSB executable, x86-64, version 1 [...]
```

- eu-readelf -Ws --elf-section can directly access
  .gnu_debugdata

```
1   ~$ eu-readelf -s zip.gdd
2
3   Symbol table [28] '.symtab' contains 202 entries:
4    82 local symbols  String table: [29] '.strtab'
5     Num:            Value  Size Type   Bind   Vis       Ndx Name
6       0: 0000000000000000     0 NOTYPE LOCAL  DEFAULT  UNDEF
7       1: 0000000000408db0   494 FUNC   LOCAL  DEFAULT     15 freeup
8       2: 0000000000408fa0  1015 FUNC   LOCAL  DEFAULT     15 DisplayRunningStats
9       3: 00000000004093a0   128 FUNC   LOCAL  DEFAULT     15 help
10  [...]
```

`help` **looks promising**[1].

```
11  ~$ eu-readelf --symbols /usr/bin/zip | grep help
12  ~$
```

---

[1]Promising as in: we may be able to trigger it with /usr/bin/zip --help.

```
1    ~$ gdb --nx --args /usr/bin/zip --help
2    Reading symbols from /usr/bin/zip...
3    Reading symbols from .gnu_debugdata for /usr/bin/zip...
4    (No debugging symbols found in .gnu_debugdata for /usr/bin/zip)
5    Missing separate debuginfos, use: dnf debuginfo-install zip-3.0-25.fc31.x86_64
6    (gdb) b help
7    Breakpoint 1 at 0x4093a0
8    (gdb) r
9    Starting program: /usr/bin/zip --help
10
11   Breakpoint 1, 0x00000000004093a0 in help ()
12   (gdb)
```

Success and two things to note:

1. Symbols read from `.gnu_debugdata`
2. No debug symbols installed for zip

---

[2]GDB 8.3 is what ships with Fedora 31

Set and hit breakpoint on `help` with LLDB 9.0.0[3]

```
1  ~$ lldb -x /usr/bin/zip -- --help
2  (lldb) target create "/usr/bin/zip"
3  Current executable set to '/usr/bin/zip' (x86_64).
4  (lldb) settings set -- target.run-args  "--help"
5  (lldb) b help
6  Breakpoint 1: no locations (pending).
7  WARNING:  Unable to resolve breakpoint to any actual locations.
8  (lldb)
```

☹ 👉 📊

---

[3]LLDB 9.0.0 is what ships with Fedora 31

⚠ Let's talk `.symtab`

Symtab

- normally, `.dynsym` is subset
- **but** for mini-debuginfo `.dynsym` symbols are stripped[4]

Implications for LLDB (and other tools)

- parse `.dynsym`
    - when no `.symtab` found **or**
    - when mini-debuginfo present and smuggled in

---
[4]https://sourceware.org/gdb/current/onlinedocs/gdb/MiniDebugInfo.html

☑ Show that LLDB can now find `help` symbol

```
1  $ lldb -x /usr/bin/zip -- --help
2  (lldb) target create "/usr/bin/zip"
3  Current executable set to '/usr/bin/zip' (x86_64).
4  (lldb) settings set -- target.run-args  "--help"
5  (lldb) b help
6  Breakpoint 1: where = zip`help, address = 0x00000000004093a0
7  (lldb) r
8  Process 277525 launched: '/usr/bin/zip' (x86_64)
9  Process 277525 stopped
10 * thread #1, name = 'zip', stop reason = breakpoint 1.1
11     frame #0: 0x00000000004093a0 zip`help
12 zip`help:
13 -> 0x4093a0 <+0>:  pushq  %r12
14    0x4093a2 <+2>:  movq   0x2af6f(%rip), %rsi       ;  + 4056
15    0x4093a9 <+9>:  movl   $0x1, %edi
16    0x4093ae <+14>: xorl   %eax, %eax
17 (lldb)
```

♡

🚢 Ready to ship?

⑦ What tests exists for mini-debuginfo?

- find symbol from `.gnu_debugdata`
- warning when decompressing `.gnu_debugdata` w/o LZMA support
- error when decompressing corrupted xz
- full example with compiled and modified code in accordance to gdb's documentation

🛌 fell asleep yet?

</> **Example test file in Shell test suite**

lldb/test/Shell/Breakpoint/example.c:

```
1   // REQUIRES: system-linux, lzma, xz
2   // RUN: gcc -g -o %t %s
3   // RUN: %t 1 2 3 4 | FileCheck %s
4
5   #include <stdio.h>
6   int main(int argc, char* argv[]) {
7
8     // CHECK: Number of {{.*}}: 5
9     printf("Number of arguments: %d\n", argc);
10
11    return 0;
12  }
```

```
~/llvm-project$ llvm-lit lldb/test/Shell/Breakpoint/example.c
-- Testing: 1 tests, 1 workers --
PASS: lldb-shell :: Breakpoint/example.c (1 of 1)

Testing Time: 0.20s
  Expected Passes    : 1
```

Thank you!

Please share your feedback ★★★★★

https://submission.fosdem.org/feedback/10393