gdb 會使用到下面兩條指令

show debug-file-directory

set debug-file-directory /usr/lib/debug

這篇是一堆試誤心得的中間記錄, 使用的版本是 Ubuntu 11.04。

失敗的作法

- 在 link 時, 用 -L/usr/lib/debug/lib/x86_64-linux-gnu/ 改變 link 到的 libc.so, 但沒有效果。用 strace -e open 觀察 gcc 做的事, 發現是因為 /usr/lib/debug/lib/x86_64-linux-gnu/ 下沒有 libc.so, 而是 libc-2.13.so。之前沒學清楚 -L 和 -l 的細節, 要笨。
- 由 man ld.so 得知可用 LD_LIBRARY_PATH 或 LD_PRELOAD 在執行期換掉 libc.so, 但是也沒有效果。用 LD_PRELOAD 換成 debug 版 libc.so 時, 跑 gdb 會 segmentation fault

成功的作法

前置作業

- $ sudo aptitude install libc6-dbg # 取得有 debug symbol 的 libc.so
- $ apt-get source libc6-dev # 取得原始碼目錄 eglibc-2.13

執行

1. $ gcc myprog.c -g -o myprog
2. $ gdb myprog
3. $ directory /path/to/eglibc-2.13/stdio-common/
4. $ start # 跑到 main 就停下來

然後 gdb 會神奇地去找含 debug symbol 的 libc, 後面就可以用 step 進入 glibc 的函式。不知這個行為寫在那裡, 或許可以從 gdb 原始碼找出來吧。

若有進入但說找不到原始碼, 表示沒有告知 gdb 正確的原始碼位置, 到 eglibc-2.13 下找一找, 再回來用 directory 設位置。

另外在用到 sqrt()、log() 時也是如此, 照一樣的編法 gcc myprog.c -g -lm -o myprog, 然後在 start 後, gdb 會去找 debug 版的 libm.so。不過要記得多執行 directory /path/to/eglibc-2.13/math 載入 math 的原始碼, gdb 才能列出原本的程式。用 ldd 觀察 myprog 也驗證原本的執行檔將 libc 和 libm 連到沒有 debug symbol 的版本。

另外試了直接和 debug 版的 libc.so 或 libm.so 編在一起 (gcc myprog.c -g /usr/lib/debug/lib/x86_64-linux-gnu/libc-2.13.so), 但是一跑就會 segmentation fault。

另外有些函式好像是用組語寫的, 看不懂它們的行為, step into sqrt 沒有效果。

結論

要觀察 glibc 的行為, 要做以下的事:

- 裝 libc6-dbg, 取得含 debug symbol 的 shared lib
- 用 apt-get source libc6-dev 取得原始碼。由於 glibc 裡有多個 shared lib, 要先 grep 找看看想觀察的程式放在那個目錄下, 跑 gdb 時再用 directory 載入該目錄, 相對路徑才會對。

2012-01-10 更新: 補充觀察 gdb 找 debug lib 的行為

一樣可以用老招 strace -e open 跑 gdb 看出背後發生的事, 以下是沒有裝 libc6-dbg 跑出的訊息:

```
$ strace -e open -o gdb.trace gdb myprog
```

然後執行 tail -f gdb.trace | grep libc 觀察行為。
以下是執行 start 以前的訊息:

```
1   open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY) = 3
2   open("/lib/libcrypto.so.0.9.8", O_RDONLY) = 3
3   open("/usr/share/locale/en_US.UTF-8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT
4   (No such file or directory)
5   open("/usr/share/locale/en_US.utf8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT
6   (No such file or directory)
7   open("/usr/share/locale/en_US/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No
8   such file or directory)
9   open("/usr/share/locale/en.UTF-8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No
10  such file or directory)
11  open("/usr/share/locale/en.utf8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No
12  such file or directory)
13  open("/usr/share/locale/en/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such
14  file or directory)
    open("/usr/share/locale-langpack/en_US.UTF-8/LC_MESSAGES/libc.mo", O_RDONLY) =
    -1 ENOENT (No such file or directory)
    open("/usr/share/locale-langpack/en_US.utf8/LC_MESSAGES/libc.mo", O_RDONLY) = -1
    ENOENT (No such file or directory)
    open("/usr/share/locale-langpack/en_US/LC_MESSAGES/libc.mo", O_RDONLY) = -1
    ENOENT (No such file or directory)
```

```
open("/usr/share/locale-langpack/en.UTF-8/LC_MESSAGES/libc.mo", O_RDONLY) = -1
ENOENT (No such file or directory)
open("/usr/share/locale-langpack/en.utf8/LC_MESSAGES/libc.mo", O_RDONLY) = -1
ENOENT (No such file or directory)
open("/usr/share/locale-langpack/en/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT
(No such file or directory)
```

以下是執行 start 以後的訊息:

```
1  open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY) = 6
2  open("/lib/x86_64-linux-gnu/libc-2.13.so", O_RDONLY) = 7
3  open("/lib/x86_64-linux-gnu/.debug/libc-2.13.so", O_RDONLY) = -1 ENOENT (No such
4  file or directory)
5  open("/usr/lib/debug//lib/x86_64-linux-gnu/libc-2.13.so", O_RDONLY) = -1 ENOENT
6  (No such file or directory)
7  open("/usr/lib/debug/lib/x86_64-linux-gnu/libc-2.13.so", O_RDONLY) = -1 ENOENT
8  (No such file or directory)
   open("/lib/x86_64-linux-gnu/libc-2.13.so-gdb.py", O_RDONLY) = -1 ENOENT (No such
   file or directory)
   open("/usr/lib/debug/lib/x86_64-linux-gnu/libc-2.13.so-gdb.py", O_RDONLY) = -1
   ENOENT (No such file or directory)
   open("/usr/share/gdb/auto-load/lib/x86_64-linux-gnu/libc-2.13.so-gdb.py",
   O_RDONLY) = -1 ENOENT (No such file or directory)
```

可以看出 gdb 不論如何, 都會試著載入 debug 版的函式庫, 來執行目標程式。找不到的時候, 自然就是用沒有 debug symbol 的函式庫。