使用 gdb 调试 glibc

我在学习 glibc 在 main 函数之前的时候，其实已经有类似的需求了，只不过当时不够激进，没有想到调试 glibc 。 这两天在学习 NPTL 线程栈及 TLS 相关的东西，实在是比较复杂，才动了调试 glibc，单步跟踪一窥究竟的念头。在网上找了一些资料，解决了这个问题。中间遇到的很有意思的东西比较有价值的东西我都记录下来了，这篇文章不能算原创，基本来源于参考文献的两篇文章。向这两位同学致谢。

glibc 查看版本号

1 ldd 某可执行程序 查看链接情况

2 直接执行链接的 libc.so 文件 。

```
root@manu:~/code/c/self/tls#
root@manu:~/code/c/self/tls# ldd test2
        linux-gate.so.1 =>  (0xb771c000)
        libpthread.so.0 => /lib/i386-linux-gnu/libpthread.so.0 (0xb76e4000)
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb753b000)
        /lib/ld-linux.so.2 (0xb771d000)
root@manu:~/code/c/self/tls# ll /lib/i386-linux-gnu/libc.so.6
lrwxrwxrwx 1 root root 12  1月 28 20:30 /lib/i386-linux-gnu/libc.so.6 -> libc-2.15.so*
root@manu:~/code/c/self/tls# /lib/i386-linux-gnu/libc-2.15.so
GNU C Library (Ubuntu EGLIBC 2.15-0ubuntu10.4) stable release version 2.15, by Roland McGrath et al.
Copyright (C) 2012 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
Compiled by GNU CC version 4.6.3.
Compiled on a Linux 3.2.35 system on 2013-01-28.
Available extensions:
        crypt add-on version 2.1 by Michael Glad and others
        GNU Libidn by Simon Josefsson
        Native POSIX Threads Library by Ulrich Drepper et al
        BIND-8.2.3-T5B
libc ABIs: UNIQUE IFUNC
For bug reporting instructions, please see:
<http://www.debian.org/Bugs/>.
root@manu:~/code/c/self/tls#
```

兄弟们可能会纳闷，SO 文件怎么可以执行呢。 GLIBC 耍了个 trick。

```
#ifdef HAVE_ELF
/* This function is the entry point for the shared object.
   Running the library as a program will get here. */
extern void __libc_main (void) __attribute__ ((noreturn));
void
__libc_main (void)
{
  __libc_print_version ();
  _exit (0);
}
#endif
static const char banner[] =
"GNU C Library "PKGVERSION RELEASE" release version "VERSION", by Roland McGrath et
al.\n\
Copyright (C) 2012 Free Software Foundation, Inc.\n\
This is free software; see the source for copying conditions.\n\
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A\n\
PARTICULAR PURPOSE.\n\
Compiled by GNU CC version "__VERSION__".\n"
```

```
...
void
__libc_print_version (void)
{
  __write (STDOUT_FILENO, banner, sizeof banner - 1);
}
```

glibc 应该是和 Ubuntu 没啥关系，可是我们从上图中居然看到 Ubuntu EGLIBC 的字样，原因很简单，Ubuntu 发行给 glibc-2.15 打了 patch。

这个问题的解决，要感谢 CSDN 的 blog of tony 。再次向这位前辈致敬。

gdb 调试 *glibc*

原本查看版本好的目的是为了下载对应源码，开始用 gdb 调试 glibc，可是上面截图中的 Ubuntu EGLIBC 字样给我泼了一盆冷水。何哉？我虽然有 glibc-2.15 的 source code，无奈 Ubuntu 给 glibc 打了 patch。 继续找办法。

在 Ubuntu 下，首先是安装符号表。 我们知道，大多数的动态库 DSO ，都是 strip 过的，没有任何调试信息，第一步是要给 glibc 添加符号表。保存在/usr/lib/debug/对应目录下。

第一步 安装符号表

```
sudo apt-get install libc6-dbg
```

安装完后，我们在可以看到如下：

```
root@manu:/usr/lib/debug/lib/i386-linux-gnu# ll
总用量 10268
drwxr-xr-x 3 root root    4096  5月  2 16:21 ./
drwxr-xr-x 4 root root    4096 12月  2 15:16 ../
-rwxr-xr-x 1 root root  555181 1月  28 20:30 ld-2.15.so*
-rw-r--r-- 1 root root   49332 1月  28 20:30 libanl-2.15.so
-rw-r--r-- 1 root root   13544 1月  28 20:30 libBrokenLocale-2.15.so
-rwxr-xr-x 1 root root 6963435 1月  28 20:30 libc-2.15.so*
-rw-r--r-- 1 root root   62923 1月  28 20:30 libcidn-2.15.so
-rw-r--r-- 1 root root   68453 1月  28 20:30 libcrypt-2.15.so
...
-rw-r--r-- 1 root root    6149 1月  28 20:30 libpcprofile.so
-rwxr-xr-x 1 root root  565440 1月  28 20:30 libpthread-2.15.so*
-rw-r--r-- 1 root root  201563 1月  28 20:30 libresolv-2.15.so
-rw-r--r-- 1 root root  135143 1月  28 20:30 librt-2.15.so
....
```

不太清楚为何某人安装/usr/lib/debug/lib/i386-linux-gnu 的筒子可以阅读我的另一篇博文：程序减肥，strip，eu-strip 及其符号表

这仅仅是有了符号表，但是看不到代码，调试的过程中，无法看到代码。很难受。

第二步：安装源码：

Ubuntu 很贴心的提供了安装方法：

```
root@manu:~/code/c/classical# sudo apt-get source libc6-dev
```

在我的当前路径下下载好了源码包，patch 包，最终生成了 eglibc-2.15 路径。所有源码都在该目录下：

drwxr-xr-x 72 root root 4096 5 月 2 17:05 eglibc-2.15/

精确的源码我们也有了，现在可以用 gdb 来调试 glibc 了，终于走到了这一步：

第三步：**gdb 调试 glibc**

我的目的是通过调试 pthread_create 创建过程，了解 glibc 和内核是如何实现 NPTL 线程的。 我的程序里面有 pthread_create，我想单步跟踪 glibc 里面的 __pthread_create_2_1 函数。

首先要告诉的 gdb，源文件要去那个目录下寻找：

(gdb) directory /home/manu/code/c/classical/eglibc-2.15/nptl

OK，我们可以看下截图：

```
Reading symbols from /home/manu/code/c/self/tls/test2...done.
(gdb) directory /home/manu/code/c/classical/eglibc-2.15/nptl
Source directories searched: /home/manu/code/c/classical/eglibc-2.15/nptl:$cdir:$cwd
(gdb) b pthread_create
Breakpoint 1 at 0x8048590
(gdb) r
Starting program: /home/manu/code/c/self/tls/test2
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".
hello
 I am the main , GS address b7dfa700

Breakpoint 1, 0x08048590 in pthread_create@plt ()
(gdb) s
Single stepping until exit from function pthread_create@plt,
which has no line number information.
__pthread_create_2_1 (newthread=0xbffff6d4, attr=0x0, start_routine=0x8048689 <child1>, arg=0x0) at pthread_create.c:452
452     {
(gdb) n
456       if (iattr == NULL)
(gdb) n
462       int err = ALLOCATE_STACK (iattr, &pd);
(gdb) l
457         /* Is this the best idea?  On NUMA machines this could mean
458            accessing far-away memory.  */
459         iattr = &default_attr;
460
461       struct pthread *pd = NULL;
```

这部分内容，主要参考自[用 GDB 追踪 glibc 代码执行过程](#)。感谢作者的分享，让我可以继续我的 NPTL 线程栈和 TLS 的分析。希望近期能完成这个 NPTL 的学习。

值得一提的是，分析 Ubuntu 提供的 patch，可以看到，glibc 查看版本号一节中的输出中含有 Ubuntu EGLIC 的原因：

-rw-r--r-- 1 root root 10368576 1 月 28 19:03 eglibc_2.15-0ubuntu10.4.diff

```
+   $(call logme, -a $(log_build), echo -n "Build started: " ; date --rfc-2822 ; echo "---------------") ; \
+   $(call logme, -a $(log_build), \
+       cd $(DEB_BUILDDIR) && \
+       CC="$(call xx,CC)" \
+       CXX="$(call xx,CXX)" \
+       AUTOCONF=false \
+       $(CURDIR)/configure \
+       --host=$(call xx,configure_target) \
+       --build=$$configure_build --prefix=/usr --without-cvs \
+       --enable-add-ons=$(standard-add-ons)"$(call xx,add-ons)" \
+       --enable-profile \
+       --without-selinux \
+       --enable-stackguard-randomization \
+       --with-pkgversion="Ubuntu EGLIBC $(DEB_VERSION)" \
+       --with-bugurl="http://www.debian.org/Bugs/" \
+       $(call xx,with_headers) $(call xx,extra_config_options))
+   touch $@
```

参考文献

1 用 GDB 追踪 glibc 代码执行过程

2 如何查看 GLIBC 的版本