【建议收藏】数据库源码学习调试利器之 CGDB

爱可生开源社区 2024-10-25 126 阅读6分钟

‹‹‹ TRAE 2.0 SOLO 出道，一键贯通从灵感火花到上线部署的全程协作 ›››

CGDB 是 GDB 的一个前端工具，通过提供更丰富的界面来增强 GDB 的用户体验。如果更喜欢在增强型终端中操作，可以使用 CGDB 来代替 GDB。

> 作者：赵黎明，爱可生 MySQL DBA 团队成员，熟悉 Oracle、MySQL 等数据库，擅长数据库性能问题诊断、事务与锁问题的分析等，负责处理客户 MySQL 及我司自研 DMP 平台日常运维中的问题，对开源数据库相关技术非常感兴趣。 爱可生开源社区出品，原创内容未经授权不得随意使用，转载请联系小编并注明来源。

本文约 2000 字，预计阅读需要 10 分钟。

简介

CGDB (Curses-based GDB)：是一个基于文本界面的 GDB 前端，主要用于在终端中提供更丰富的用户界面，CGDB 使用 Curses 库 创建了一个简单的功能界面，帮助用户更方便地使用 GDB，它在 GDB 的基础上增加了一些功能，使得调试过程更加直观和高效。

CGDB 的运行依赖 GDB 环境，因此，在调试前必须先安装符合其版本要求的 GDB

简单来说，**CGDB 是 GDB 的一个前端工具，通过提供更丰富的界面来增强 GDB 的用户体验**。如果更喜欢在增强型终端中操作，可以使用 CGDB 来代替 GDB。

版本选择

本次选择安装 *gdb 9.2* 的版本，原因主要有以下两个：

1. CentOS 7.5 中自带 gdb 的版本 7.6.1-120.el7，而 cgdb 要求 gdb 版本为 7.12 及以上。
2. 安装 gdb 9.0 以上版本的，还可以用于调试 OBServer，否则会报版本错误。

安装 CGDB

安装步骤：

bash 体验AI代码助手 代码解读 复制代码

```bash
-- 安装依赖
yum -y install automake flex texinfo ncurses-devel readline-devel gcc-c++

-- 下载源码包
git clone https://github.com/cgdb/cgdb.git

-- 编译源码
cd cgdb
./autogen.sh
./configure --prefix=/usr/local
make && make install
```

如果在执行 make 时报错：`error: 'for' loop initial declarations are only allowed in C99 mode`，可在进行编译配置时加上参数：`CFLAGS="-std=c99"`，如：`CFLAGS="-std=c99" ./configure --prefix=/usr/local`。

爱可生开源社区 LV.5
一个有深度的数据库社区 @上...
榜上有名 优秀作者
234 文章 249k 阅读 177 粉丝
关注 私信

目录 收起

相关推荐
最新java学习资料集合（建议收藏）
538阅读 · 1点赞
《Go学习路线图》让你少走弯路，Let's...
57k阅读 · 659点赞
深度学习模型训练基础环境搭建有效整...
615阅读 · 7点赞
Flutter学习资料集合(入门进阶必备，建...
2.4k阅读 · 13点赞
如何使用cgdb + qemu调试linux内核模块
1.1k阅读 · 4点赞

精选内容
TCP三次握手的智慧：为什么不是两次...
有才叔 · 69阅读 · 2点赞
限流算法百科全书：从原理到实践，...
都叫我大帅哥 · 45阅读 · 0点赞
当RAG学会"思考"：Agentic RAG架构完...
都叫我大帅哥 · 26阅读 · 0点赞
Rust 实战四 | Traui2+Vue3+Rspack 开...
集成显卡 · 48阅读 · 1点赞
Maven在使用过程中的核心知识点总结
熊猫片沃子 · 53阅读 · 0点赞

```
pen.c: In function 'vterm_state_newpen':
pen.c:166:3: error: 'for' loop initial declarations are only allowed in C99 mode
   for(int col = 0; col < 16; col++)
   ^
pen.c:166:3: note: use option -std=c99 or -std=gnu99 to compile your code
make[3]: *** [pen.o] Error 1
make[3]: Leaving directory `/data/coredump/cgdb/lib/vterm'
make[2]: *** [all-recursive] Error 1
make[2]: Leaving directory `/data/coredump/cgdb/lib'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/data/coredump/cgdb'
make: *** [all] Error 2
02:52 PM dmp3 {master} /data/coredump/cgdb#
```


安装 GDB

注意事项：

- 尽量不装 10.x 及以上的高版本。可能会报错：`A compiler with support for C++11 language features is required`。
- CentOS 7.5 默认的 gcc 版本较低（4.8.5），原则上只要够用就行，没必要追求高版本。

安装步骤：

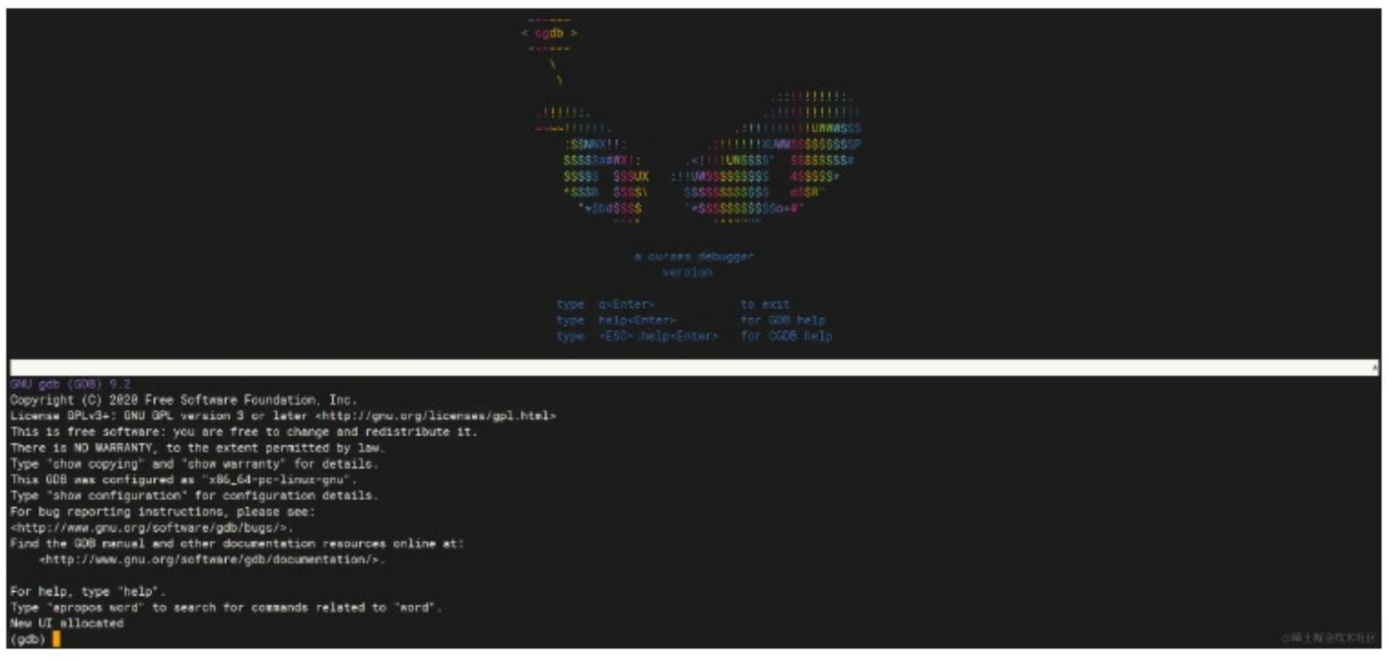
bash 体验AI代码助手 代码解读 复制代码

```bash
-- 安装依赖
yum -y install gcc gcc-c++ texinfo

-- 下载源码包
wget ftp://ftp.gnu.org/gnu/gdb/gdb-9.2.tar.gz

-- 解压并编译
tar zxf gdb-9.2.tar.gz -C /tmp
cd /tmp/gdb-9.2
mkdir build && cd build
/tmp/gdb-9.2/configure
make && make install
```

查看 gdb 的版本，确认是否升级成功。

```
01:51 PM dmp3 /tmp/gdb-9.2/build# gdb -v
GNU gdb (GDB) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
01:51 PM dmp3 /tmp/gdb-9.2/build#
```


执行 cgdb，进入调试界面。



查看帮助

- 键入 `help + 回车键`，可查看所有的 gdb 的指令和说明
- 键入 `ESC + :help + 回车键`，可查看所有 cgdb 的指令和说明

具体指令和说明不在文中展示。

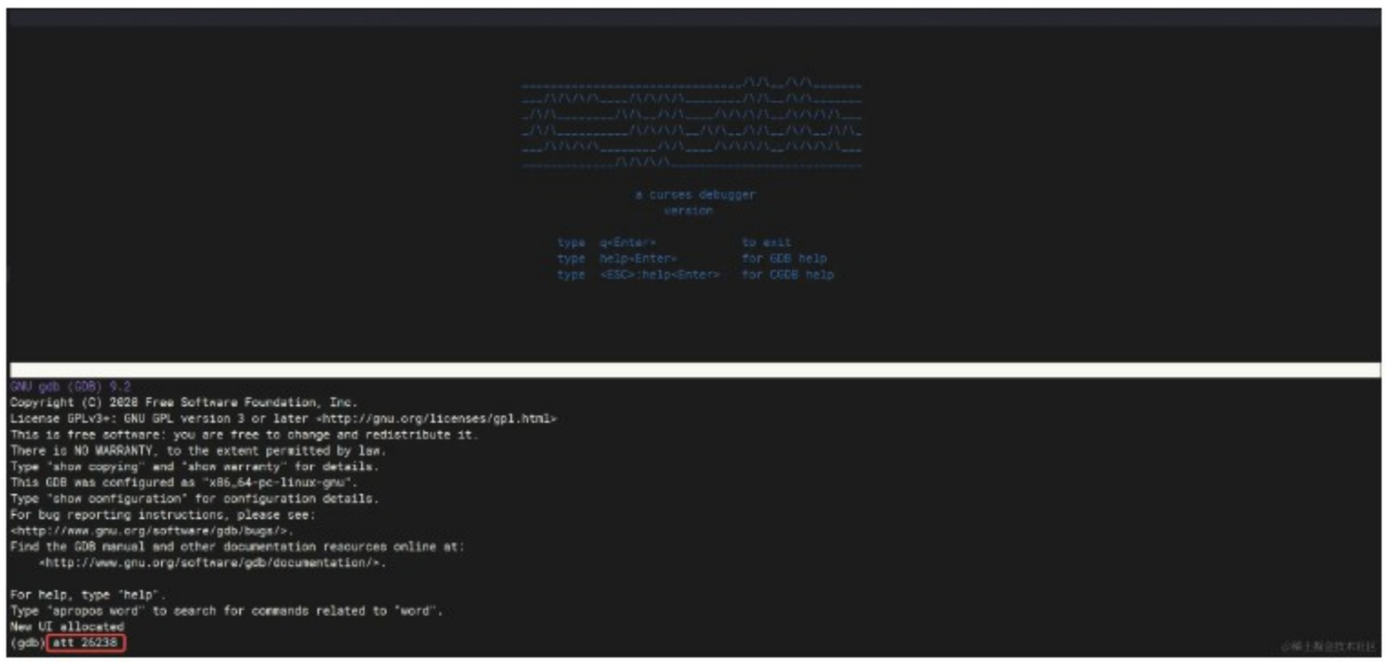
下面我们通过几个常用的场景示例，演示 CGDB 和 GDB 的使用过程和效果。

调试示例

示例 1：调试 MySQL 获取源码

查看 *mysqld* 的进程号，此处为 26238。

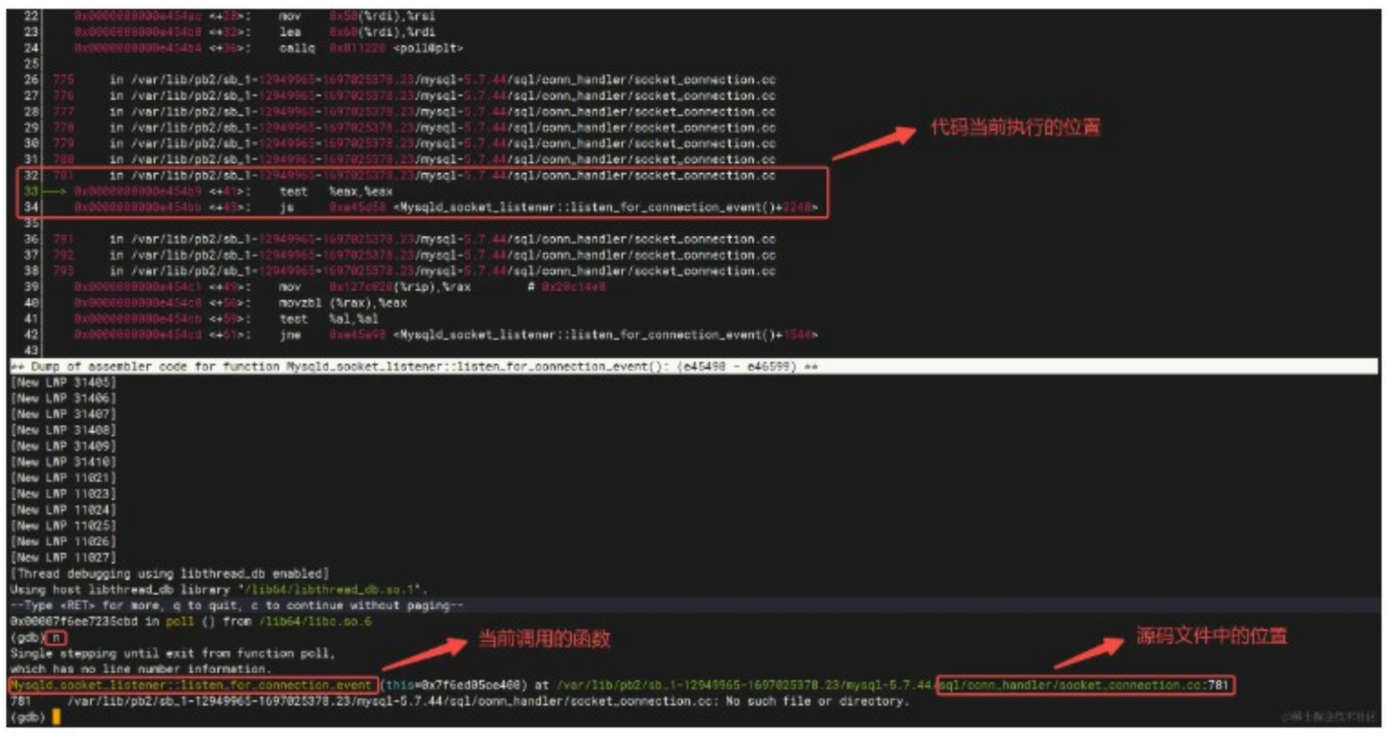
在 gdb 窗口执行 `att 26238`，将其 attach 到 mysqld 进程上。



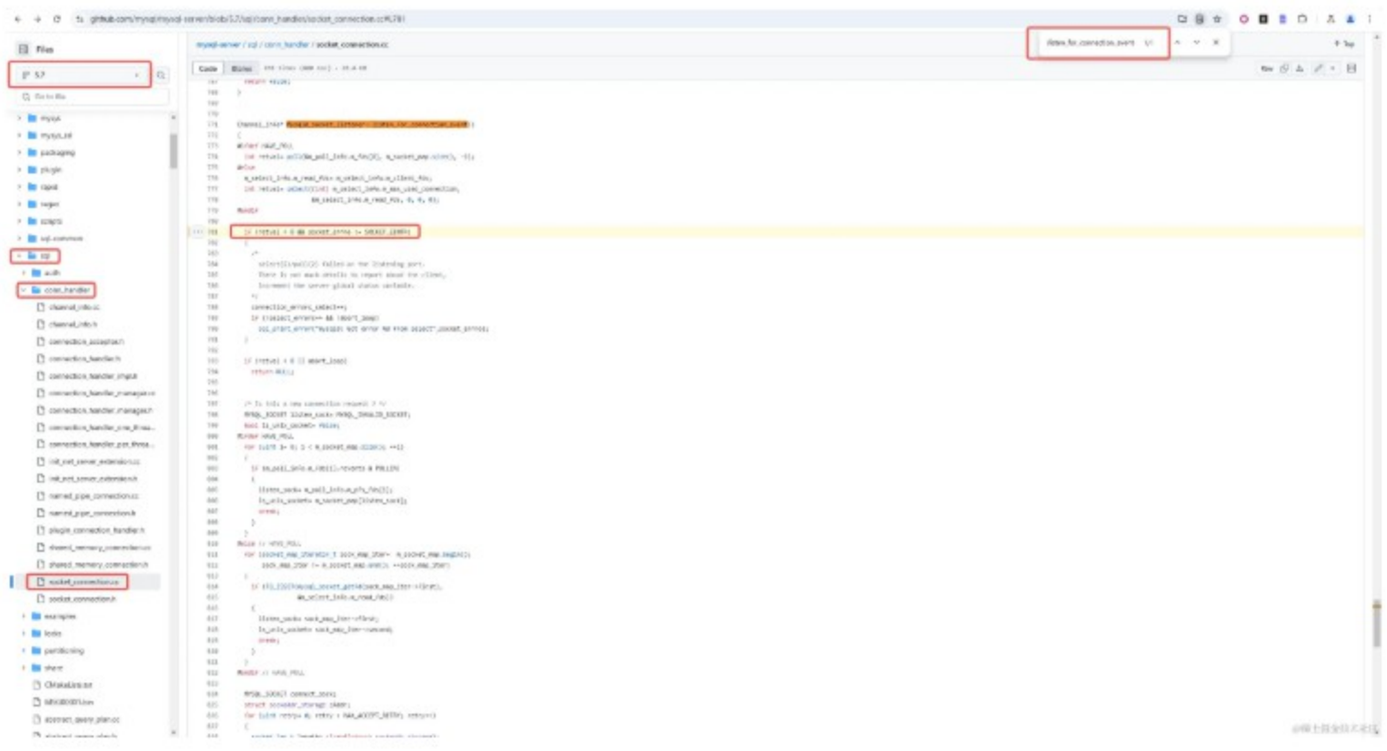
绿色箭头代表代码当前执行的位置，会展示代码所处行号，内存地址，代码文件等信息。

按 ESC 键，会进入上半部分的代码展示窗口，能像在 vim 中那样用快捷键上下移动光标进行查看。

如果要返回 gdb 的窗口，按 i 键即可，就能继续执行调试命令了。



根据打印的源码文件和位置，去官网代码库中找到对应的文件，再搜索相应的函数，就可以获取对应的源码内容了。



示例 2：调试 MySQL 线程

执行 `info threads`，打印所有线程。

```
** Dump of assembler code for function poll: (7fd63f0feca0 - 7fd63f0fecf9) **
  Id   Target Id                                  Frame
* 1    Thread 0x7fd640f02fc0 (LWP 960) "mysqld"  0x00007fd63f0feccd in poll () from /lib64/libc.so.6
  2    Thread 0x7fd6336ec700 (LWP 964) "mysqld"  0x00007fd63f04258a in sigwaitinfo () from /lib64/libc.so.6
  3    Thread 0x7fd627c83700 (LWP 965) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  4    Thread 0x7fd627482700 (LWP 966) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  5    Thread 0x7fd626c81700 (LWP 967) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  6    Thread 0x7fd626480700 (LWP 968) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  7    Thread 0x7fd625c7f700 (LWP 969) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  8    Thread 0x7fd62547e700 (LWP 970) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  9    Thread 0x7fd624c7d700 (LWP 971) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  10   Thread 0x7fd62447c700 (LWP 972) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  11   Thread 0x7fd623c7b700 (LWP 973) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  12   Thread 0x7fd62347a700 (LWP 974) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  13   Thread 0x7fd622c79700 (LWP 975) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  14   Thread 0x7fd621fff700 (LWP 981) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  15   Thread 0x7fd6211ff700 (LWP 982) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  16   Thread 0x7fd6209fe700 (LWP 983) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  17   Thread 0x7fd6201fd700 (LWP 984) "mysqld"  0x00007fd640658e9d in nanosleep () from /lib64/libpthread.so.0
  18   Thread 0x7fd61f9fc700 (LWP 985) "mysqld"  0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  19   Thread 0x7fd61f1fb700 (LWP 986) "mysqld"  0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  20   Thread 0x7fd61e9fa700 (LWP 987) "mysqld"  0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
--Type <RET> for more, q to quit, c to continue without paging--
```

依次执行 `thread`、`bt`，查看当前线程及该线程的 backtrace。

```
** Dump of assembler code for function poll: (7fd63f0feca0 - 7fd63f0fecf9) **
--Type <RET> for more, q to quit, c to continue without paging--
  189  Thread 0x7fd6043a3700 (LWP 2287) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  190  Thread 0x7fd604371700 (LWP 2288) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  191  Thread 0x7fd60433f700 (LWP 2289) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  192  Thread 0x7fd60430d700 (LWP 2290) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  193  Thread 0x7fd6042db700 (LWP 2291) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  194  Thread 0x7fd6042a9700 (LWP 2292) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  195  Thread 0x7fd604277700 (LWP 2293) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  196  Thread 0x7fd604245700 (LWP 2294) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  197  Thread 0x7fd604213700 (LWP 2295) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
(gdb) thread
[Current thread is 1 (Thread 0x7fd640f02fc0 (LWP 960))]
(gdb) bt
#0  0x00007fd63f0feccd in poll () from /lib64/libc.so.6
#1  0x0000000000e454b9 in Mysqld_socket_listener::listen_for_connection_event (this=0x7fd6285a4a00)
    at /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc:774
#2  0x0000000000822280 in Connection_acceptor<Mysqld_socket_listener>::connection_event_loop (this=0x7fd6397ce4b8)
    at /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/connection_acceptor.h:73
#3  mysqld_main (argc=126, argv=0x7fd63ea3de88) at /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/mysqld.cc:5156
#4  0x00007fd63f02d555 in __libc_start_main () from /lib64/libc.so.6
#5  0x0000000000815699 in _start ()
(gdb)
```

从下往上

多次执行 `s`，一行一行地进行单步调试，注意调试期间 thread 是否发生变化。

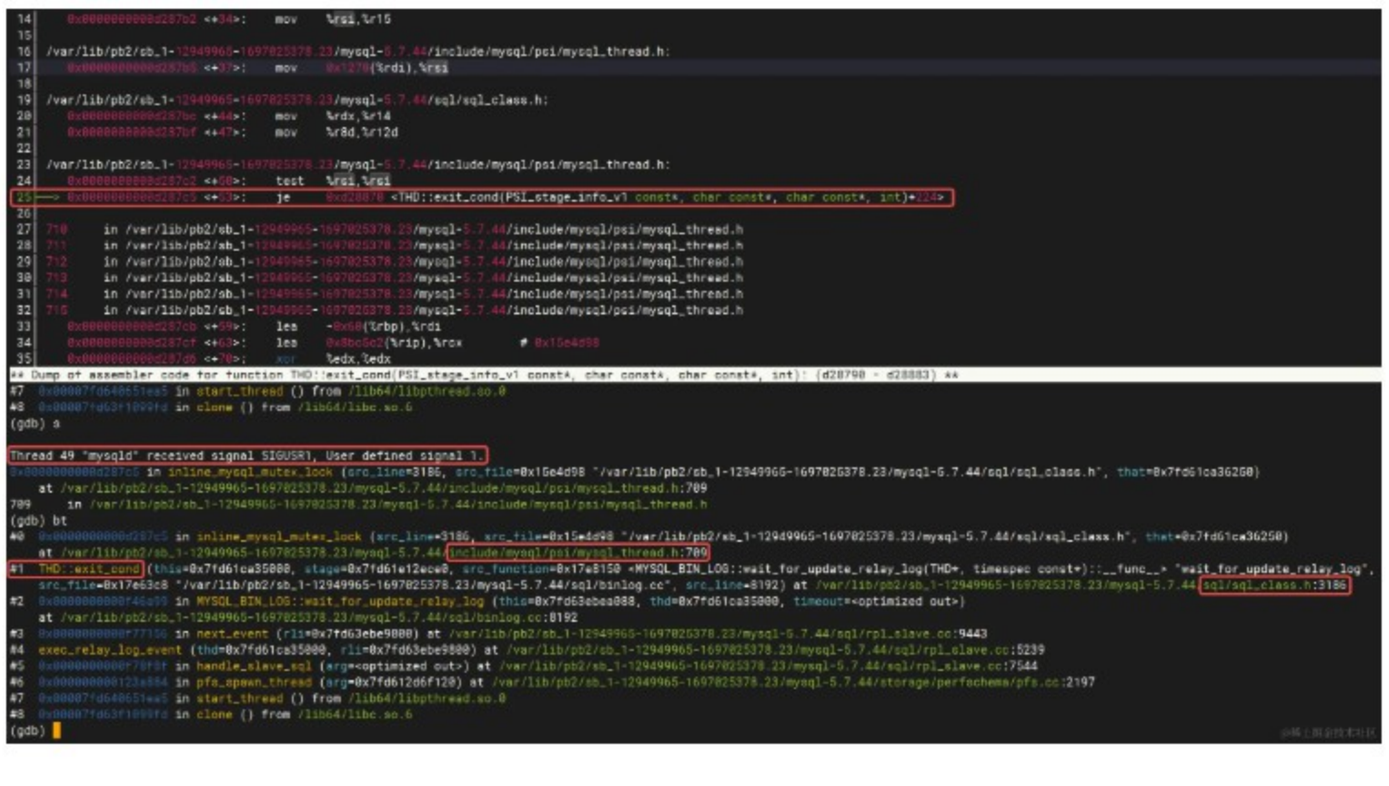
```
** Dump of assembler code for function Mysqld_socket_listener::listen_for_connection_event(): (e45490 - e46599) **
Single stepping until exit from function poll,
which has no line number information.
Mysqld_socket_listener::listen_for_connection_event (this=0x7fd6285a4a00) at /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc:781
781     /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc: No such file or directory.
(gdb) s
793     in /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc
(gdb) s
801     in /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc
(gdb) s
std::map<st_mysql_socket, bool, Socket_lt_type, std::allocator<std::pair<st_mysql_socket const, bool> > >::size (this=0x7fd6285a4a28)
    at /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc:801
801     in /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc
(gdb) s
std::_Rb_tree<st_mysql_socket, std::pair<st_mysql_socket const, bool>, std::_Select1st<std::pair<st_mysql_socket const, bool> >, Socket_lt_type, std::allocator<std::pair<st_mysql_socket const,
bool> > >::size (this=0x7fd6285a4a28) at /usr/lib/gcc/x86_64-redhat-linux/4.4.7/../../../../include/c++/4.4.7/bits/stl_tree.h:671
671     /usr/lib/gcc/x86_64-redhat-linux/4.4.7/../../../../include/c++/4.4.7/bits/stl_tree.h: No such file or directory.
(gdb) s
Mysqld_socket_listener::listen_for_connection_event (this=0x7fd6285a4a00) at /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc:801
801     /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc: No such file or directory.
(gdb) thread
[Current thread is 1 (Thread 0x7fd640f02fc0 (LWP 960))]
(gdb)
```

当前为 ID 1 的线程，如果要切换到某个 thread，可以执行 thread [thread_id] 进行切换。

```
** Dump of assembler code for function Mysqld_socket_listener::listen_for_connection_event(): (e45490 - e46599) **
  Id   Target Id                                  Frame
* 1    Thread 0x7fd640f02fc0 (LWP 960) "mysqld"  Mysqld_socket_listener::listen_for_connection_event (this=0x7fd6285a4a00)
    at /var/lib/pb2/sb_1-12949965-1697025378.23/mysql-5.7.44/sql/conn_handler/socket_connection.cc:801
  2    Thread 0x7fd6336ec700 (LWP 964) "mysqld"  0x00007fd63f04258a in sigwaitinfo () from /lib64/libc.so.6
  3    Thread 0x7fd627c83700 (LWP 965) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  4    Thread 0x7fd627482700 (LWP 966) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  5    Thread 0x7fd626c81700 (LWP 967) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  6    Thread 0x7fd626480700 (LWP 968) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  7    Thread 0x7fd625c7f700 (LWP 969) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  8    Thread 0x7fd62547e700 (LWP 970) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  9    Thread 0x7fd624c7d700 (LWP 971) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  10   Thread 0x7fd62447c700 (LWP 972) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  11   Thread 0x7fd623c7b700 (LWP 973) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  12   Thread 0x7fd62347a700 (LWP 974) "mysqld"  0x00007fd640244644 in __io_getevents_0_4 () from /lib64/libaio.so.1
  13   Thread 0x7fd622c79700 (LWP 975) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  14   Thread 0x7fd621fff700 (LWP 981) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  15   Thread 0x7fd6211ff700 (LWP 982) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  16   Thread 0x7fd6209fe700 (LWP 983) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  17   Thread 0x7fd6201fd700 (LWP 984) "mysqld"  0x00007fd640658e9d in nanosleep () from /lib64/libpthread.so.0
  18   Thread 0x7fd61f9fc700 (LWP 985) "mysqld"  0x00007fd640655de2 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  19   Thread 0x7fd61f1fb700 (LWP 986) "mysqld"  0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
--Type <RET> for more, q to quit, c to continue without paging--
```

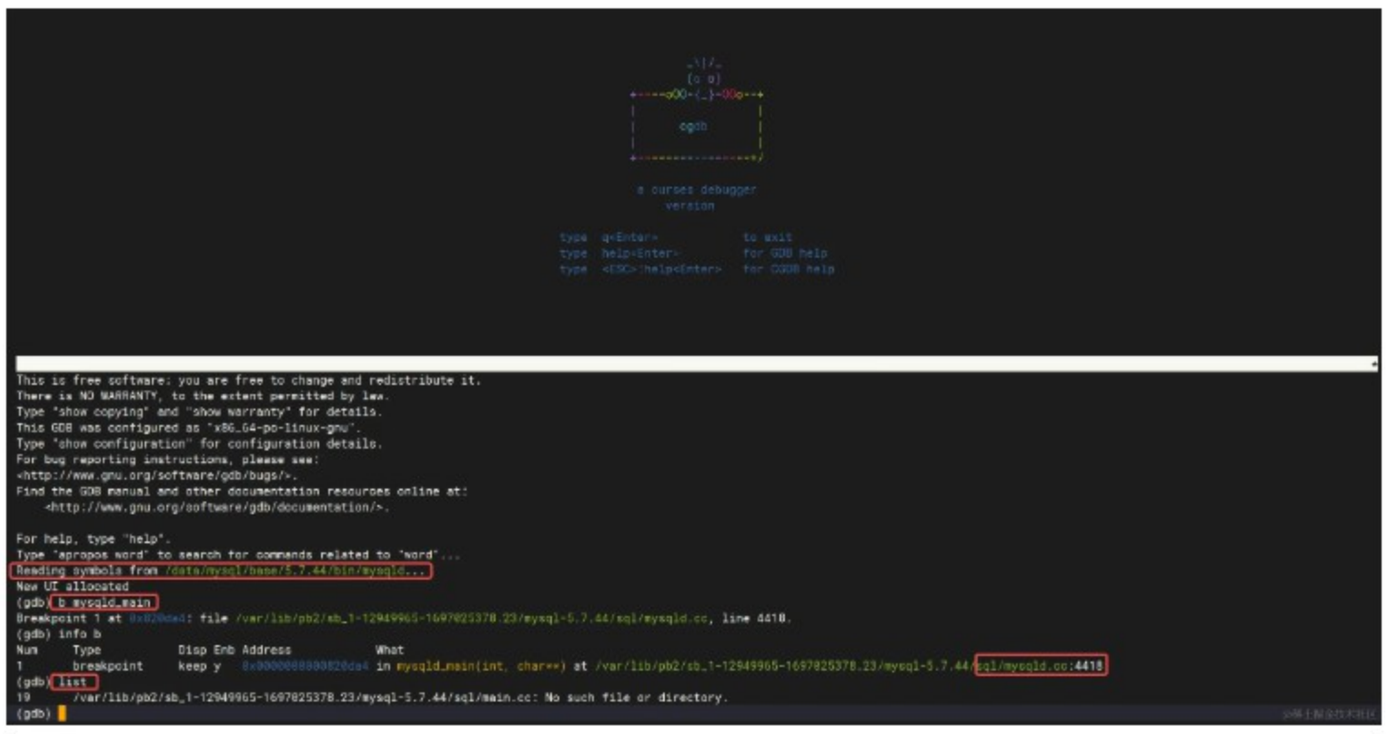
```
** Dump of assembler code for function pthread_cond_wait@@GLIBC_2.3.2: (7fd640655970 - 7fd640655b92) **
  185  Thread 0x7fd60476b700 (LWP 2283) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  186  Thread 0x7fd604439700 (LWP 2284) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  187  Thread 0x7fd604407700 (LWP 2285) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
--Type <RET> for more, q to quit, c to continue without paging--
  188  Thread 0x7fd6043d5700 (LWP 2286) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  189  Thread 0x7fd6043a3700 (LWP 2287) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  190  Thread 0x7fd604371700 (LWP 2288) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  191  Thread 0x7fd60433f700 (LWP 2289) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  192  Thread 0x7fd60430d700 (LWP 2290) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  193  Thread 0x7fd6042db700 (LWP 2291) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  194  Thread 0x7fd6042a9700 (LWP 2292) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  195  Thread 0x7fd604277700 (LWP 2293) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  196  Thread 0x7fd604245700 (LWP 2294) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  197  Thread 0x7fd604213700 (LWP 2295) "mysqld" 0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
(gdb)
(gdb)
(gdb) thread 197
[Switching to thread 197 (Thread 0x7fd604213700 (LWP 2295))]
#0  0x00007fd640655a35 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
(gdb) thread
[Current thread is 197 (Thread 0x7fd604213700 (LWP 2295))]
(gdb)
```

以下是 49 号线程打印的 backtrace 信息示例，可获取函数调用的顺序、调用的函数名、函数出现在源码文件中的位置。



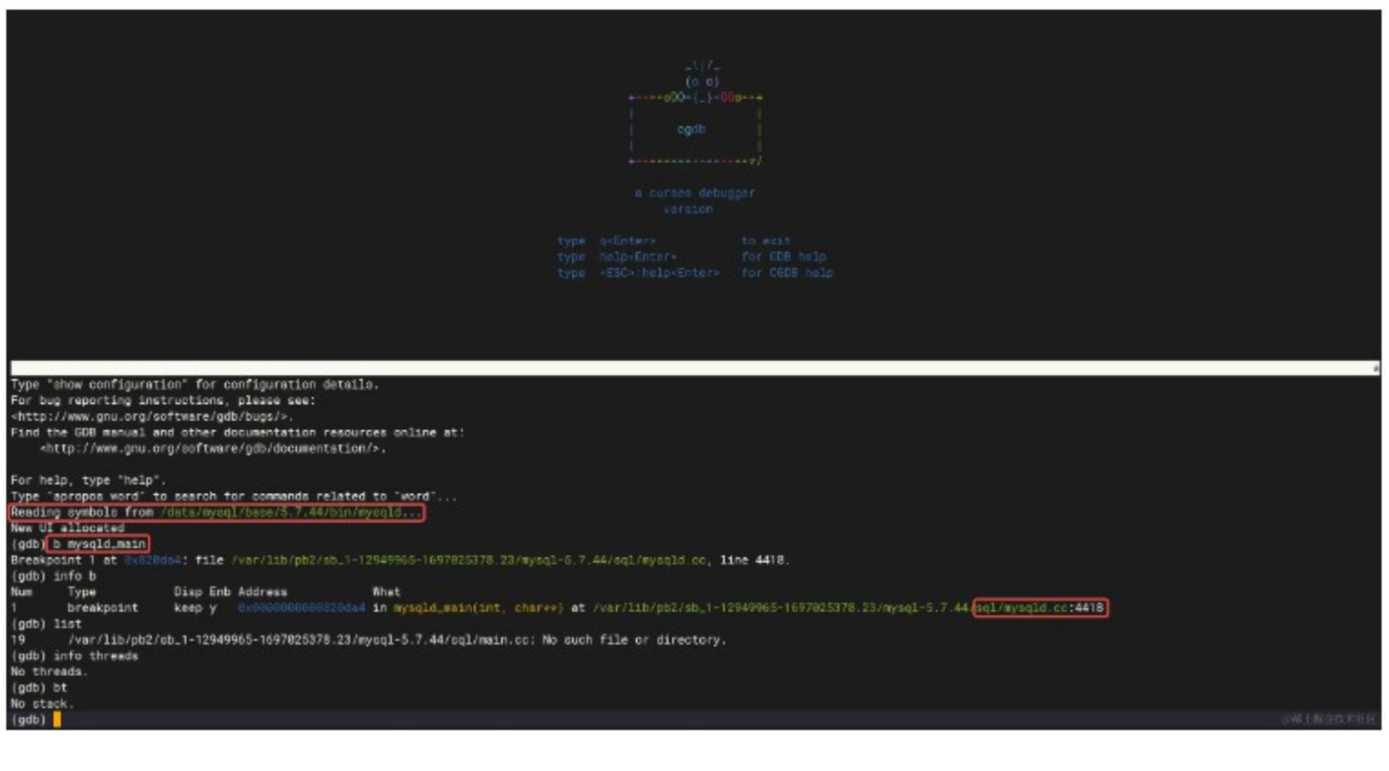
示例 3：使用 cgdb ./mysqld 调试

采用此方式调试 mysqld 时，当其还未被 attach 到 mysqld 上时，并不会阻塞新的连接。



此时只能设置断点，查看某个函数在源码文件中的位置。

由于没有线程及其帧栈信息，并不能做进一步的调试。



示例 4：分析 coredump 文件

当程序异常崩溃时，如果配置过 coredump，就可以通过分析 coredump 文件来排查程序崩溃的原因。

第一个 coredump，是通过执行 `kill -SIGSEGV [pid]` （也就是 signal 11）将 mysqld 进程杀死后产生的，其实从文件命名上就可获知，mysqld-11 后面紧跟的这个 11，就是对应信号量的编号。

```
06:12 PM dmp3 /data/coredump# ll
total 1764972
-rw------- 1 mysql mysql  949575680 Sep  9 18:19 core-mysqld-11-5702-5702-23470-dmp3-1725877183
-rw------- 1 mysql mysql 1186275328 Sep 11 16:03 core-mysqld-5-5702-5702-22721-dmp3-1726041819
-rw------- 1 mysql mysql 1265352704 Sep 11 15:13 core-mysqld-5-5702-5702-26238-dmp3-1726038795
-rw-r--r-- 1 root  root    39264162 Sep 11 13:16 gdb-9.2.tar.gz
drwxr-xr-x 2 root  root         203 Sep 11 12:53 rpm
06:12 PM dmp3 /data/coredump# cgdb /data/mysql/base/5.7.44/bin/mysqld core-mysqld-11-5702-5702-23470-dmp3-1725877183
06:13 PM dmp3 /data/coredump#
```

在 cgdb 中也打印了 mysqld 崩溃的原因，是收到了 SIGSEGV(11) 的信号量，即最常见的 `Segmentaion fault` 。

```
   0x00007f1b11929a7a <+10>:    lea    -0x20(%rsi),%edx
   0x00007f1b11929a7d <+13>:    mov    $0x10,%eax
   0x00007f1b11929a82 <+18>:    cmp    $0x1,%edx
   0x00007f1b11929a85 <+21>:    jbe    0x7f1b11929ac8 <pthread_kill+88>
   0x00007f1b11929a87 <+23>:    mov    %fs:0x2d4,%edi
   0x00007f1b11929a8f <+31>:    movslq %esi,%rdx
   0x00007f1b11929a92 <+34>:    movslq %ecx,%rsi
   0x00007f1b11929a95 <+37>:    mov    $0xea,%ecx
   0x00007f1b11929a9a <+42>:    movslq %edi,%rdi
   0x00007f1b11929a9d <+45>:    mov    %ecx,%eax
   0x00007f1b11929a9f <+47>:    syscall
=> 0x00007f1b11929aa1 <+49>:    mov    %rax,%rdx
   0x00007f1b11929aa4 <+52>:    xor    %eax,%eax
   0x00007f1b11929aa6 <+54>:    mov    %edx,%ecx
   0x00007f1b11929aa8 <+56>:    neg    %ecx
   0x00007f1b11929aaa <+58>:    cmp    $0xfffff000,%edx
   0x00007f1b11929ab0 <+64>:    cmova  %ecx,%eax
   0x00007f1b11929ab3 <+67>:    retq
   0x00007f1b11929ab4 <+68>:    mov    $0x3,%eax
   0x00007f1b11929ab9 <+73>:    nopl   0x0(%rax)
   0x00007f1b11929ac0 <+80>:    repz retq
End of assembler dump.
** Dump of assembler code for function pthread_kill: (7f1b11929a70 - 7f1b11929ac0) **
[New LWP 23521]
[New LWP 23508]
[New LWP 23528]
[New LWP 23499]
[New LWP 23497]
[New LWP 23529]
[New LWP 23518]
[New LWP 23536]
[New LWP 23498]
[New LWP 23522]
[New LWP 26128]
[New LWP 26121]
[New LWP 26122]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
--Type <RET> for more, q to quit, c to continue without paging--
Core was generated by `/data/mysql/base/5.7.44/bin/mysqld --defaults-file=/data/mysql/mysql3332/my.cnf'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x00007f1b11929aa1 in pthread_kill () from /lib64/libpthread.so.0
[Current thread is 1 (Thread 0x7f1b121d5fc0 (LWP 23470))]
New UI allocated
(gdb)
```

第二个 coredump，是在用 cgdb 调试时生成的，期间执行过 run 命令，将 mysqld 进程重启过，产生了 mysqld-5 的 coredump 文件。

```
06:15 PM dmp3 /data/coredump# ll
total 1764972
-rw------- 1 mysql mysql  949575680 Sep  9 18:19 core-mysqld-11-5702-5702-23470-dmp3-1725877183
-rw------- 1 mysql mysql 1186275328 Sep 11 16:03 core-mysqld-5-5702-5702-22721-dmp3-1726041819
-rw------- 1 mysql mysql 1265352704 Sep 11 15:13 core-mysqld-5-5702-5702-26238-dmp3-1726038795
-rw-r--r-- 1 root  root    39264162 Sep 11 13:16 gdb-9.2.tar.gz
drwxr-xr-x 2 root  root         203 Sep 11 12:53 rpm
06:15 PM dmp3 /data/coredump# cgdb /data/mysql/base/5.7.44/bin/mysqld core-mysqld-5-5702-5702-22721-dmp3-1726041819
```

在 cgdb 中也打印了 mysqld 崩溃的原因，是收到了 SIGTRAP(5) 的信号量。

```
   0x00007fac5c8f8dbd <+269>:   testl  $0x1,0x28(%rdi)
   0x00007fac5c8f8dc4 <+276>:   mov    $0x100,%edx
   0x00007fac5c8f8dc9 <+281>:   mov    $0xffffffff,%r9d
   0x00007fac5c8f8dcf <+287>:   cmove  %edx,%eax
   0x00007fac5c8f8dd2 <+290>:   or     %eax,%esi
   0x00007fac5c8f8dd4 <+292>:   mov    %r12,%rdx
   0x00007fac5c8f8dd7 <+295>:   add    $0x4,%rdi
   0x00007fac5c8f8ddb <+299>:   mov    $0xca,%eax
   0x00007fac5c8f8de0 <+304>:   syscall
   0x00007fac5c8f8de2 <+306>:   mov    %rax,%r14
   0x00007fac5c8f8de5 <+309>:   mov    (%rsp),%edi
=> 0x00007fac5c8f8de8 <+312>:   callq  0x7fac5c8fb4a0 <__pthread_disable_asynccancel>
   0x00007fac5c8f8ded <+317>:   mov    0x8(%rsp),%rdi
   0x00007fac5c8f8df2 <+322>:   mov    $0x1,%esi
   0x00007fac5c8f8df7 <+327>:   xor    %eax,%eax
   0x00007fac5c8f8df9 <+329>:   lock cmpxchg %esi,(%rdi)
   0x00007fac5c8f8dfd <+333>:   jne    0x7fac5c8f8f80 <pthread_cond_timedwait@@GLIBC_2.3.2+640>
   0x00007fac5c8f8e03 <+339>:   mov    0x2c(%rdi),%edx
   0x00007fac5c8f8e06 <+342>:   mov    0x18(%rdi),%rax
   0x00007fac5c8f8e0a <+346>:   mov    0x10(%rdi),%r9
   0x00007fac5c8f8e0e <+350>:   cmp    0x4(%rsp),%edx
   0x00007fac5c8f8e12 <+354>:   jne    0x7fac5c8f8e57 <pthread_cond_timedwait@@GLIBC_2.3.2+423>
** Dump of assembler code for function pthread_cond_timedwait@@GLIBC_2.3.2: (7fac5c8f8cb0 - 7fac5c8f8f6b) **
[New LWP 22780]
[New LWP 22745]
[New LWP 22747]
[New LWP 22779]
[New LWP 22746]
[New LWP 22777]
[New LWP 22764]
[New LWP 22763]
[New LWP 22754]
[New LWP 22751]
[New LWP 22750]
[New LWP 22749]
[New LWP 22748]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
--Type <RET> for more, q to quit, c to continue without paging--
Core was generated by `/data/mysql/base/5.7.44/bin/mysqld --defaults-file=/data/mysql/mysql3332/my.cnf'.
Program terminated with signal SIGTRAP, Trace/breakpoint trap.
#0  0x00007fac5c8f8de8 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
[Current thread is 1 (Thread 0x7fac19177700 (LWP 26413))]
New UI allocated
(gdb)
```

如果对信号量不太熟悉，可用 `kill -l` 命令查看，它会输出所有信号量。

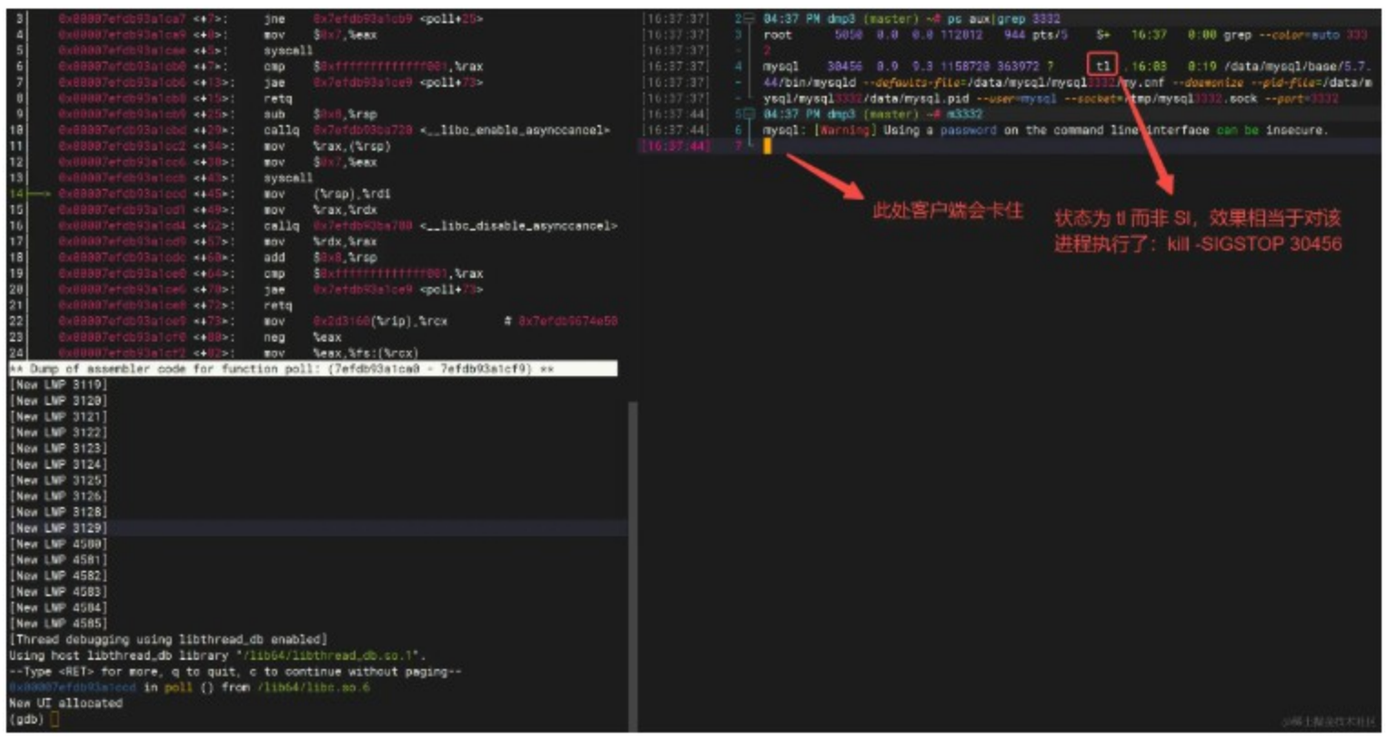
```
06:18 PM dmp3 (master) ~# kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL      10) SIGUSR1
11) SIGSEGV      12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT    17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN      22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM    27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS       34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4   39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9   44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14  49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11  54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6   59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1   64) SIGRTMAX
06:18 PM dmp3 (master) ~#
```

示例 5：使用 cgdb -p 调试

与之前先进入 cgdb 调试台，再执行 `attach [pid]` 的方式并无区别，后者会在 cgdb/gdb 进程中显示 mysqld 进程号。

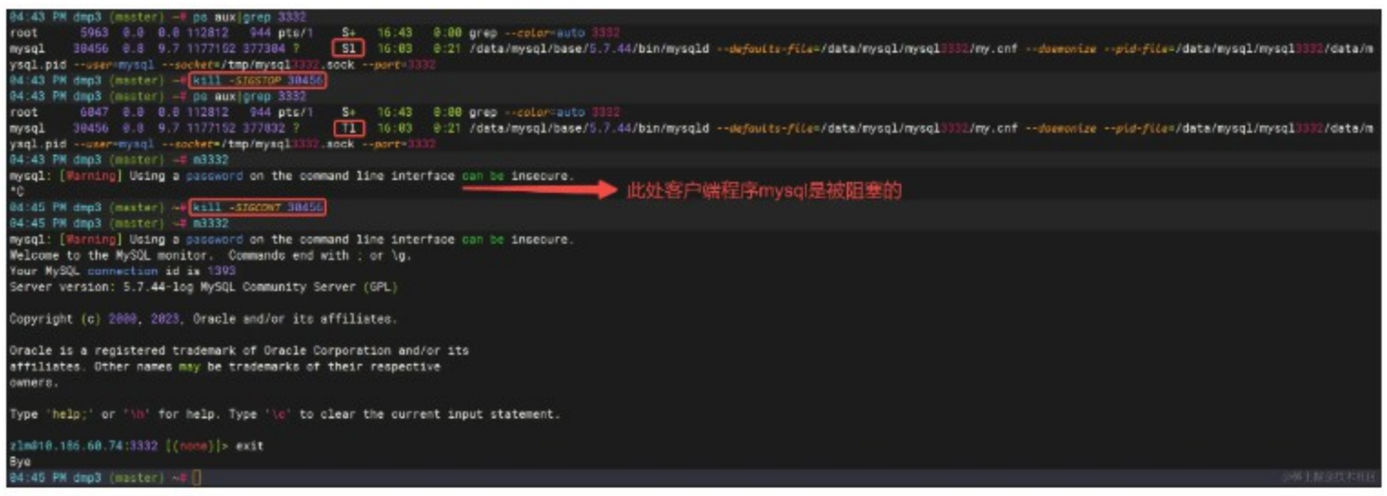
```
06:05 PM dmp3 /data/coredump# ps -ef|grep gdb
root     27079 31389  0 18:07 pts/0    00:00:00 cgdb -p 20403
root     27080 27079 99 18:07 pts/5    00:00:02 gdb --nw -ex new-ui mi /dev/pts/3 -p 20403
root     27091  5297  0 18:07 pts/1    00:00:00 grep --color=auto gdb
06:07 PM dmp3 /data/coredump# ps -ef|grep gdb
root     27232 31389  0 18:07 pts/0    00:00:00 cgdb
root     27233 27232 20 18:07 pts/5    00:00:02 gdb --nw -ex new-ui mi /dev/pts/3
root     27276  5297  0 18:07 pts/1    00:00:00 grep --color=auto gdb
```


要注意的是，这两种方式都会直接阻塞 mysql 客户端，因为此时 mysqld 会被阻塞，导致无法建立新的连接。



用 SIGSTOP/SIGCONT 的信号量来观测效果

> Tips：信号量名中的 SIG 是可以被省略的，如： `kill -SIGSTOP [pid]` 和 `kill -STOP [pid]` 是等效的。



示例 6：单独起一个 mysqld 调试

该方式可以在不影响已运行 mysqld 的基础上，对同版本的 mysqld 单独进行调试。

建议下载带 boost 的 MySQL 源码包，然后编译为 Debug 版本，可以打印更多的 debugging symbols 信息，方便调试。

```
11:11 AM dmp3 (master) ~# cgdb --args /data/mysql/base/5.7.44/bin/mysqld
11:12 AM dmp3 (master) ~#
```


示例 7：修改 MySQL 最大连接数

当 MySQL 的连接数满导致无法登陆实例时，可以用 cgdb 来救急。

下图中，当客户端连接实例时报错：`"Too many connections"`，直接用 cgdb/gdb 来调大 `max_connections` 参数的值。

如果服务器上有多个 mysqld 进程时，建议直接指定 pid，否则可能改到了另一个 MySQL 实例上。

用 cgdb 修改

```
06:29 PM dmp3 (master) ~# m3332 -se "show variables like 'max_connections'";
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1040 (HY000): Too many connections
06:29 PM dmp3 (master) ~# cgdb --pid=`pidof mysqld` -ex "set max_connections=100" --batch
06:29 PM dmp3 (master) ~# m3332 -se "show variables like 'max_connections'";
mysql: [Warning] Using a password on the command line interface can be insecure.
Variable_name   Value
max_connections 100
06:29 PM dmp3 (master) ~#
```


用 gdb 修改

```
06:54 PM dmp3 (master) ~# m3332 -se "show variables like 'max_connections'";
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1040 (HY000): Too many connections
06:54 PM dmp3 (master) ~# gdb --pid=`pidof mysqld` -ex "set max_connections=100" --batch
10194: No such file or directory.
[New LWP 31429]
[New LWP 31428]
```



```
[New LWP 12429]
[New LWP 12428]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
0x00007f7107c7bccd in poll () at ../sysdeps/unix/syscall-template.S:81
81      T_PSEUDO (SYSCALL_SYMBOL, SYSCALL_NAME, SYSCALL_NARGS)
[Inferior 1 (process 12424) detached]
06:55 PM dmp3 (master) ~# m3332 -se "show variables like 'max_connections'";
mysql: [Warning] Using a password on the command line interface can be insecure.
Variable_name   Value
max_connections 100
06:55 PM dmp3 (master) ~#
```


总结

- 本文简单介绍了 CGDB 及其基本使用方法。
- 利用 CGDB 调试工具，能帮助我们梳理程序在运行时各种函数的调用逻辑，这对于学习和研究程序源码非常有帮助。
- 当程序崩溃时，如果能拿到故障现场的 `coredump` 文件，可通过 CGDB 去分析程序崩溃的原因，如：在特定场景下，在调用某个函数时触发了程序的 bug 而引发的崩溃。
- 当 MySQL 连接数被打满后，除了我们已知的 `extra_port` 方法之外，还可以用 CGDB 来解决。
- 注意，生产环境严禁使用 CGDB 直接进行调试。

更多技术文章，请访问：opensource.actionsky.com/

关于 SQLE

SQLE 是一款全方位的 SQL 质量管理平台，覆盖开发至生产环境的 SQL 审核和管理。支持主流的开源、商业、国产数据库，为开发和运维提供流程自动化能力，提升上线效率，提高数据质量。

标签： 数据库

评论 0

登录 / 注册 即可发布评论!

暂无评论数据

为你推荐

MySQL 配置文件添加参数后服务起不来了
爱可生开源社区 | 1年前 | 1.3k 点赞 评论 数据库

Flask教程(8)--数据库操作flask_sqlalchemy
waws520 | 4年前 | 2.0k 3 评论 Flask

DataX与DB2导入导出案例
WHYBIGDATA | 2年前 | 1.0k 26 2 大数据 数据库 掘金·日...

Oracle 迁移到 OB 过程中的函数改造案例
爱可生开源社区 | 9月前 | 85 点赞 评论 数据库

MySQL 之下载安装及单张表的操作
亦黑迷失 | 1年前 | 484 1 评论 数据库 MySQL

NestJs 上手之路之二 【连接Mysql数据库、项目框架调整优化】
wd591 | 3年前 | 4.6k 25 7 Node.js

在 CentOS 8 上安装 MongoDB
Rainny | 4年前 | 2.6k 5 评论 数据库

为什么你的 show slave status 会卡住?
爱可生开源社区 | 12月前 | 176 2 评论 数据库

你要懂的的数据库知识（简单，详细）
zhouzhouya | 1年前 | 545 3 2 后端 Mongo... 数据库

深入理解 MySQL 中的 SQL_MODE
爱可生开源社区 | 11月前 | 86 点赞 评论 数据库

MySQL 如何高效可靠处理持久化数据
溪溪哈哈 | 2年前 | 509 2 评论 MySQL

前端调用数据库，使用可视化DBeaver创建数据 | 开始搞全栈（二）
董员外 | 1年前 | 644 2 评论 前端 JavaSc... 数据库

Go 操作 Mysql 数据库 | Go主题月
江景 | 4年前 | 1.8k 1 评论 Go

MongoDB从入门到实战之Windows快速安装MongoDB
追逐时光者 | 3月前 | 98 1 评论 后端 MongoDB

【egg】egg-mysql
hannie76327 | 4年前 | 3.6k 8 4 Node.js