GDB基本功能

2015年10月3日 星期六　　下午5:31

一、简介

GDB是GNU开源组织发布的一个强大的UNIX下的程序调试工具，主要用于C/C++的调试。

下面介绍如何开始调试、调试的命令、环境等常见问题、经典使用场景等。

二、开始GDB调试

三种进入GDB调试的情景

A 程序出core文件，追问题

格式：gdb 可执行文件 core文件

eg：gdb clicksim_train core.23419

B 一个正在运行的程序出问题，通过进程pid进入正在运行的程序中调试

格式：gdb -p 进程pid

或者 gdb 可执行文件 进程pid

eg：gdb -p 22315 或者 gdb clicksim_train 22315

C 用GDB起一个程序，模拟程序的执行过程，调试

格式：gdb 可支持文件

eg: gdb clicksim_train

另外还有一种方式：

先进如gdb环境，然后载入可执行文件

先在命令行直接gdb

进入到gdb环境中，输入：file可执行文件

注意，执行到这里，程序并没有开始运行，需要执行r(run)，让程序运行，详见"常用功能"中

注意：

A 编译的时候，编译选项要加-g

B 编译的时候，编译选项不能加-O3优化

三、常用功能

1、bt

查看core文件的问题，常用bt看栈内存情况。

格式：bt

2、list

查看源代码

List 如果不加任何参数 表示 显示当前行后面的源程序

格式：list / list 行号 /list 函数名

3、r(run)

在“用GDB起一个程序，模拟程序的执行过程”这种使用方式下，run让程序真正运行起来。对应二、中的情景C

格式：r / run

4、b(break)

设置断点，在运行中的程序遇到断点就会停下（对应二、中的情景B、C）

格式：b 函数名 / b 代码行数 / break 函数名 /break 代码行数

5、c(continue)

继续运行，直到下一个断点或者程序结束（对应二、中的情景B、C）

格式：c/continue

6、n(next)、s(step)

单步运行（对应二、中的情景B、C）

next和step都是单步执行，唯一的区别是：如果有函数调用，step会步入函数中，而next不会

格式：n/next

7、p(print)

打印指定变量的变量值

格式：p 变量名

1、作用域

在GDB中，你可以随时查看以下三种变量的值：

A 全局变量（所有文件可见的）

B 静态全局变量（当前文件可见的）

C 局部变量（当前Scope可见的）

file::variable
function::variable
例如 ：(gdb) p 'f2.c'::x

2、打印数组变量
p *array@len
“@”的左边是第一个内存的地址的值，“@”的右边则是你想查看内存的长度
例如：

```
(gdb) p *array@len
$1 = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24,
```

3、显示合适的数据类型格式

```
x   按十六进制格式显示变量。
d   按十进制格式显示变量。
u   按十六进制格式显示无符号整型。
o   按八进制格式显示变量。
t   按二进制格式显示变量。
a   按十六进制格式显示变量。
c   按字符格式显示变量。
f   按浮点数格式显示变量。
```

```
(gdb) p i
$21 = 101
(gdb) p/a i
$22 = 0x65
(gdb) p/c i
$23 = 101 'e'
(gdb) p/f i
$24 = 1.41531145e-43
(gdb) p/x i
$25 = 0x65
(gdb) p/t i
$26 = 1100101
```

8、查看信息

info local/info locals
查看局部变量值
info signals
info handle
查看有哪些信号在被GDB检测中
info frame/info f
打印出更为详细的当前栈层的信息
info args
打印出当前函数的参数名及其值。
info catch
打印出当前的函数中的异常处理信息
info registers

info registers

查看寄存器的情况

9、设置变量值

更改程序中变量的变量值（对应二、中的情景B、C）

格式：set 变量名 = 变量值

10、附：（一篇博客中整理的常用功能list）

命令	描述
backtrace（或bt）	查看各级函数调用及参数
finish	连续运行到当前函数返回为止，然后停下来等待命令
frame（或f） 帧编号	选择栈帧
info（或i） locals	查看当前栈帧局部变量的值
list（或l）	列出源代码，接着上次的位置往下列，每次列10行
list 行号	列出从第几行开始的源代码
list 函数名	列出某个函数的源代码
next（或n）	执行下一行语句
print（或p）	打印表达式的值，通过表达式可以修改变量的值或者调用函数
quit（或q）	退出gdb调试环境
set var	修改变量的值
start	开始执行程序，停在main函数第一行语句前面等待命令
step（或s）	执行下一行语句，如果有函数调用则进入到函数中

四、一些经典使用场景

1、单步调试、设置断点并查看信息

场景：

不论是在程序正在运行中，我们通过进程pid进入到正在运行的程序中看信息，还是重头开始run一个程序，都可以用单步调试看一些变量等信息。

用法：

如果是进入到正在运行的程序中，直接跳过A、B两步，进入C步

A 运行

如果是重头开始run，则输入r，把程序运行起来；

B 暂停

如果是重头开始run，则在运行中ctrl+c，让程序暂停

C 设置断点

List 看一下代码，看一下现在运行到哪里了

然后break，设置断点

D 继续运行

c(continue)直接运行到断点处

或者 n(next)单步

或者 s(step)单步

E 看信息

Info local

p 变量名

bt 看栈空间

set XXX 改变变量值

五、设置环境变量、库等（用于情景C）

有些时候用不了，需要把外部的环境变量、库、参数等添加进来。

1、函数输入参数

set args 可指定运行时参数。（如：set args 10 20 30 40 50）

show args 命令可以查看设置好的运行参数。

2、运行环境

path <dir> 可设定程序的运行路径。

show paths 查看程序的运行路径。

set environment varname [=value] 设置环境变量。如：set env USER=hchen

show environment [varname] 查看环境变量。

3、工作目录

cd <dir> 相当于shell的cd命令。

pwd 显示当前的所在目录。

4、环境变量

set $环境变量 = 环境变量值

show convenience

该命令查看当前所设置的所有的环境变量

六、高级功能

一些功能一般用不到，在这里提一下，知道有这个功能就行，用到的话现查。

线程相关信息、不同语言、跳转执行、信号量、历史记录、维护断点(停止点？)

七、实验环境

work@nj01-nlp-test01.nj01.baidu.com:/home/work/tianzhiliang/test/cpp/gdb

gdb_test.cpp

gdb_test.sh

八、参考资料

1、比较全面详细，但是思路层次有点乱

http://wiki.ubuntu.org.cn/%E7%94%A8GDB%E8%B0%83%E8%AF%95%E7%A8%8B%E5%BA%8F

2、有个常用功能的表格，有详细的单步调试过程

http://www.cnblogs.com/hankers/archive/2012/12/07/2806836.html