陈皓专栏 【空谷幽兰，心如皓月】

芝兰生于深谷，不以无人而不芳；君子修道立德，不为困穷而改节。

用GDB调试程序（七）

分类： 编程工具 2003-07-12 16:42 12062人阅读 评论(7) 收藏 举报

改变程序的执行

一旦使用GDB挂上被调试程序，当程序运行起来后，你可以根据自己的调试思路来动态地在GDB中更改当前被调试程序的运行线路或是其变量的值，这个强大的功能能够让你更好的调试你的程序，比如，你可以在程序的一次运行中走遍程序的所有分支。

一、修改变量值

修改被调试程序运行时的变量值，在GDB中很容易实现，使用GDB的print命令即可完成。如：

```
(gdb) print x=4
```

x=4这个表达式是C/C++的语法，意为把变量x的值修改为4，如果你当前调试的语言是Pascal，那么你可以使用Pascal的语法：x:=4。

在某些时候，很有可能你的变量和GDB中的参数冲突，如：

```
(gdb) whatis width
type = double
(gdb) p width
$4 = 13
(gdb) set width=47
Invalid syntax in expression.
```

因为，set width是GDB的命令，所以，出现了“Invalid syntax in expression”的设置错误，此时，你可以使用set var命令来告诉GDB，width不是你GDB的参数，而是程序的变量名，如：

```
(gdb) set var width=47
```

另外，还可能有些情况，GDB并不报告这种错误，所以保险起见，在你改变程序变量取值时，最好都使用set var格式的GDB命令。

二、跳转执行

一般来说，被调试程序会按照程序代码的运行顺序依次执行。GDB提供了乱序执行的功能，也就是说，GDB可以修改程序的执行顺序，可以让程序执行随意跳跃。这个功能可以由GDB的jump命令来完：

(62817)
再谈“我是怎么招聘程序员
(54256)
一些重要的算法
(51996)
C++ 对象的内存布局（上
(46037)
哥是玩程序的
(43844)
清华大学出版社“抄袭事件
(43597)
恐怖的C++语言
(39271)

评论排行

清华大学出版社“抄袭事件 (323)
再谈“我是怎么招聘程序员 (264)
“清华大学出版社抄袭事件 (233)
我是怎么招聘程序员的 (233)
C++ 虚函数表解析 (203)
6个变态的C语言写的Hel (201)
恐怖的C++语言 (198)
优秀程序员的十个习惯 (165)
Java构造时成员初始化的 (146)
惹恼程序员的十件事 (130)

推荐文章

最新评论

跟我一起写 Makefile（一）
wangjilebeishang: 我的神啊，可以，顶《《《《《学习

五个方法成为更好的程序员
雨中风铃_: 受教！

C++ 虚函数表解析
Walkerinwind: 第一个例子结束实际运行"经"果如下或许应该是结果?

“抄袭事件”判决书
fengyishang: 很遗憾，时隔七年，我才看到七年前这一起关于著作权的纠纷；但很幸运，因为最后的结果正是我们想要的，虽然...

C++ 虚函数表解析
zhiweiarm: 受益匪浅，学习了

22条经典的编程引言
雨中风铃_: 太经典了！

C++ 对象的内存布局（下）
Huang850615: cout << " ";
pFun(); //D::f();这个地方出错了，把它改成这样：cout...

五个方法成为更好的程序员
冰冻牙膏: 寻找不同观点，取长补短

跟我一起写 Makefile（一）
luomuxiaoxiao98: Mark！

其实Unix很简单
lovelock: @yhmhappy2006:呵呵，我真不知道你干了什么。。。。

jump <linespec>

指定下一条语句的运行点。<linespce>可以是文件的行号，可以是file:line格式，可以是+num这种偏移量格式。表式着下一条运行语句从哪里开始。

jump <address>

这里的<address>是代码行的内存地址。

注意，jump命令不会改变当前的程序栈中的内容，所以，当你从一个函数跳到另一个函数时，当函数运行完返回时进行弹栈操作时必然会发生错误，可能结果还是非常奇怪的，甚至于产生程序Core Dump。所以最好是同一个函数中进行跳转。

熟悉汇编的人都知道，程序运行时，有一个寄存器用于保存当前代码所在的内存地址。所以，jump命令也就是改变了这个寄存器中的值。于是，你可以使用“set $pc”来更改跳转执行的地址。如：

```
set $pc = 0x485
```

三、产生信号量

使用singal命令，可以产生一个信号量给被调试的程序。如：中断信号Ctrl+C。这非常方便于程序的调试，可以在程序运行的任意位置设置断点，并在该断点用GDB产生一个信号量，这种精确地在某处产生信号非常有利程序的调试。

语法是：signal <singal>，UNIX的系统信号量通常从1到15。所以<singal>取值也在这个范围。

single命令和shell的kill命令不同，系统的kill命令发信号给被调试程序时，是由GDB截获的，而single命令所发出一信号则是直接发给被调试程序的。

四、强制函数返回

如果你的调试断点在某个函数中，并还有语句没有执行完。你可以使用return命令强制函数忽略还没有执行的语句并返回。

return

return <expression>

使用return命令取消当前函数的执行，并立即返回，如果指定了<expression>，那么该表达式的值会被认作函数的返回值。

五、强制调用函数

call <expr>

表达式中可以一是函数，以此达到强制调用函数的目的。并显示函数的返回值，如果函数返回值是void，那么就不显示。

另一个相似的命令也可以完成这一功能——print，print后面可以跟表达式，所以也可以用他来调用函数，print和call的不同是，如果函数返回void，call则不显示，print则显示函数返回值，并把该值存入历史数据中。

在不同语言中使用GDB

GDB支持下列语言：C，C++，Fortran，PASCAL，Java，Chill，assembly，和 Modula-2。一般说来，GDB会根据你所调试的程序来确定当然的调试语言，比如：发现文件名后缀为“.c”的，GDB会认为是C程序。文件

名后缀为“.C, .cc, .cp, .cpp, .cxx, .c++”的，GDB会认为是C++程序。而后缀是“.f, .F”的，GDB会认为是Fortran程序，还有，后缀为如果是“.s, .S”的会认为是汇编语言。

也就是说，GDB会根据你所调试的程序的语言，来设置自己的语言环境，并让GDB的命令跟着语言环境的改变而改变。比如一些GDB命令需要用到表达式或变量时，这些表达式或变量的语法，完全是根据当前的语言环境而改变的。例如C/C++中对指针的语法是*p，而在Modula-2中则是p^。并且，如果你当前的程序是由几种不同语言一同编译成的，那到在调试过程中，GDB也能根据不同的语言自动地切换语言环境。这种跟着语言环境而改变的功能，真是体贴开发人员的一种设计。

下面是几个相关于GDB语言环境的命令：

```
show language
    查看当前的语言环境。如果GDB不能识为你所调试的编程语言，那么，C语言被认为是默认的环境。

info frame
    查看当前函数的程序语言。

info source
    查看当前文件的程序语言。
```

如果GDB没有检测出当前的程序语言，那么你也可以手动设置当前的程序语言。使用set language命令即可做到。

当set language命令后什么也不跟的话，你可以查看GDB所支持的语言种类：

```
    (gdb) set language
    The currently understood settings are:

    local or auto    Automatic setting based on source file
    c                Use the C language
    c++              Use the C++ language
    asm              Use the Asm language
    chill            Use the Chill language
    fortran          Use the Fortran language
    java             Use the Java language
    modula-2         Use the Modula-2 language
    pascal           Use the Pascal language
    scheme           Use the Scheme language
```

于是你可以在set language后跟上被列出来的程序语言名，来设置当前的语言环境。

后记

——

GDB是一个强大的命令行调试工具。大家知道命令行的强大就是在于，其可以形成执行序列，形成脚本。UNIX下的软件全是命令行的，这给程序开发提代供了极大的便利，命令行软件的优势在于，它们可以非常容易的集成在一起，使用几个简单的已有工具的命令，就可以做出一个非常强大的功能。

于是UNIX下的软件比Windows下的软件更能有机地结合，各自发挥各自的长处，组合成更为强劲的功能。而Windows下的图形软件基本上是各自为营，互相不能调用，很不利于各种软件的相互集成。在这里并不是要和Windows做个什么比较，所谓“寸有所长，尺有所短”，图形化工具还是有不如命令行的地方。（看到这句话时，希望各位千万再也不要认为我就是“鄙视图形界面”，和我抬杠了 ）

我是根据版本为5.1.1的GDB所写的这篇文章，所以可能有些功能已被修改，或是又有更为强劲的功能。而且，我写得非常仓促，写得比较简略，并且，其中我已经看到有许多错别字了（我用五笔，所以错字让你看不懂），所以，我在这里对我文中的差错表示万分的歉意。

文中所罗列的GDB的功能时，我只是罗列了一些带用的GDB的命令和使用方法，其实，我这里只讲述的功能大约只占GDB所有功能的60%吧，详细的文档，还是请查看GDB的帮助和使用手册吧，或许，过段时间，如果我有空，我再写一篇GDB的高级使用。

我个人非常喜欢GDB的自动调试的功能，这个功能真的很强大，试想，我在UNIX下写个脚本，让脚本自动编译我的程序，被自动调试，并把结果报告出来，调试成功，自动checkin源码库。一个命令，编译带着调试带着checkin，多爽啊。只是GDB对自动化调试目前支持还不是很成熟，只能实现半自动化，真心期望着GDB的自动化调试功能的成熟。

如果各位对GDB或是别的技术问题有兴趣的话，欢迎和我讨论交流。本人目前主要在UNIX下做产品软件的开发，所以，对UNIX下的软件开发比较熟悉，当然，不单单是技术，对软件工程实施，软件设计，系统分析，项目管理我也略有心得。欢迎大家找我交流，（QQ是：753640，MSN是：haoel@hotmail.com）

<-上一页

分享到：

上一篇：用GDB调试程序（四）
下一篇：用GDB调试程序（六）

查看评论

7楼 syzcch 2012-11-21 11:09发表

很多高级功能都没有使用过

6楼 n_anhai 2012-04-17 22:42发表

GDB真强大啊！！
之前一直用printf效率太低了。

5楼 Bu319209 2011-10-20 14:38发表

受益了，期待新作品，感谢指导！！

4楼 insulted 2009-07-27 10:39发表

very good!
我是在网上见到别人提供的两篇word技术文档下载：
《用GDB调试程序》和《跟我一起写 Makefile》，文末都作者提到了自己的兴趣点，一路baidu过来的，呵呵，踩踩！
发现楼上的finlinden朋友可真是痴情啊，呵呵，其实人在江湖，身不由己的事情时有发生，汝又何必强求呢？
不过话也说回来，我也是很期待博主将这句“过段时间，我再写一篇GDB的高级使用”话付诸于实践，也不负六年来读过此文又期待下文的朋友。
祝愿博主身体健康，工作顺利！

3楼 jznhljg 2008-11-30 01:52发表

真的讲的很不错...谢谢楼主. 期待高级应用..理论+实践=精典

2楼 finlinden 2007-10-20 15:40发表

还记得：那是2003年的第一场雨，比以往来得更晚一些，
那时，他曾语重心长地对我说：
【过段时间，我再写一篇GDB的高级使用。】
-
四年，一段不短的时间，它足可以让一个懵懂少年读完本科
我等了你四年！四年！就是在等一个机会。
我要争口气，不是要证明我了不起，我要告诉别人，如果我要等一样别人许诺过我的东西，我一定会等到亲手拿到它，才会走开！~”。

对了，等你真要著文以飨众生的时候，别忘了加上几个关于GDB高级使用的你觉得满意的case，这样例举说明，效果会更好些……

1楼 finlinden 2007-09-14 17:19发表

【或许，过段时间，如果我有空，我再写一篇GDB的高级使用。】
请问先生：何时会出一篇【GDB的高级使用】？
泣盼：早日出炉啊~！！
我现在最想看到的就是您这种有丰富实践经验的人写的GDB调试指南。
座中泣下谁最多？——等您文章的我。

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场