原 Linux系统之下的基本gdb调试

2017年10月21日 20:03:48 帅气的羊羊杨 阅读数：257 标签： linux 调试 gdb 更多

一、调试的工具： gdb

二、调试的对象： 可执行程序，而不是.c文件，调试的对象一定是一个进程。调试时，程序必须带有调试信息，所以在编译链接的过程中，需要加上参

三、调试的命令：

l 显示代码
回车 继续执行上一操作
b + 行号 加断点
info + b(break) 查看断点信息
r 运行程序
n 下一步，单步执行
p + 变量名 打印变量的值(p + &变量名 打印变量的地址)
q 退出调试
display + 变量名 持续打印
c 继续执行
delete + 断电编号 删除断点
s 进入函数
finish 跳出函数
bt 函数调用栈关系(**在被调用函数里看能看见自己和调用函数，在调用函数里只能看见自己**)

四、实例

1.先编写一个简单的c语言程序——main.c。

内容见下：

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int arr[6] = {0};
  int i=0;
  int a=0;
  int b=0;

  for( ; i<6; i++)
  {
    arr[i] = i;
    printf("%d\n",arr[i]);
  }


  exit(0);
}
```

像下面的“gcc -o main main.c”,是无法生成有调试信息的main的，因此无法进行调试。

```
[TLDS@localhost text1021]$ gcc -o main main.c
[TLDS@localhost text1021]$ gdb main
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-56.el6)
Copyright (C) 2010 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.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/TLDS/exercise/text1021/main...(no debugging symbol
ound)...done.
(gdb) l
No symbol table is loaded.  Use the "file" command.
(gdb)
```

加上参数“-g”，才可以进行调试。

```
[TLDS@localhost text1021]$ gcc -o main main.c -g
[TLDS@localhost text1021]$ ./main
0
1
2
3
4
5
[TLDS@localhost text1021]$ gdb main
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-56.el6)
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.h
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying
and "show warranty" for details.
This GDB was configured as "i686-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/TLDS/exercise/text1021/main...done.
(gdb) l
1       #include <stdio.h>
2       #include <stdlib.h>
3
4       int main()
```

2.正式开始调试：

打印全部的代码

```
(gdb) l
1       #include <stdio.h>
2       #include <stdlib.h>
3
4       int main()
5       {
6         int arr[6] = {0};
7         int i=0;
8         int a=0;
9         int b=0;
10
(gdb)
11        for( ; i<6; i++)
12        {
13          arr[i] = i;
14          printf("%d\n",arr[i]);
15        }
16
17
18        exit(0);
19      }
```

下断点、查看断点信息

```
(gdb) b 13
Breakpoint 1 at 0x804843c: file main.c, line 13.
(gdb) b 14
Breakpoint 2 at 0x8048448: file main.c, line 14.
(gdb) info b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x0804843c in main at main.c:13
2       breakpoint     keep y   0x08048448 in main at main.c:14
(gdb)
```


运行程序

```
(gdb) r
Starting program: /home/TLDS/exercise/text1021/main

Breakpoint 1, main () at main.c:13
13          arr[i] = i;
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.80.el6.i
(gdb)
```


下一步，查看变量i、arr[i]的值

```
(gdb) n

Breakpoint 2, main () at main.c:14
14          printf("%d\n",arr[i]);
(gdb) p i
$1 = 0
(gdb) p arr[i]
$2 = 0
(gdb)
```


持续显示变量i、arr[i]的值

```
(gdb) display i arr[i]
A syntax error in expression, near `arr[i]'.
(gdb) display i
1: i = 0
(gdb) diaplay arr[i]
Undefined command: "diaplay".  Try "help".
(gdb) display i
2: i = 0
(gdb) display arr[i]
3: arr[i] = 0
(gdb) n
0
11        for( ; i<6; i++)
3: arr[i] = 0
2: i = 0
1: i = 0
(gdb)

Breakpoint 1, main () at main.c:13
13          arr[i] = i;
3: arr[i] = 0
2: i = 1
1: i = 1
(gdb)
```


继续执行

```
(gdb) c
Continuing.
1

Breakpoint 1, main () at main.c:13
13          arr[i] = i;
3: arr[i] = 0
2: i = 2
1: i = 2
(gdb) c
Continuing.

Breakpoint 2, main () at main.c:14
14          printf("%d\n",arr[i]);
3: arr[i] = 2
2: i = 2
1: i = 2
(gdb)
Continuing.
2
```


删除断点

```
(gdb) info b
No breakpoints or watchpoints.
(gdb) b 13
Breakpoint 3 at 0x804843c: file main.c, line 13.
(gdb) b 14
Breakpoint 4 at 0x8048448: file main.c, line 14.
(gdb) info b
Num     Type           Disp Enb Address    What
3       breakpoint     keep y   0x0804843c in main at main.c:13
4       breakpoint     keep y   0x08048448 in main at main.c:14
(gdb) delete 1
No breakpoint number 1.
(gdb) info b
Num     Type           Disp Enb Address    What
3       breakpoint     keep y   0x0804843c in main at main.c:13
4       breakpoint     keep y   0x08048448 in main at main.c:14
(gdb) delete 3
(gdb) info b
Num     Type           Disp Enb Address    What
4       breakpoint     keep y   0x08048448 in main at main.c:14
(gdb)
```

注意删除断点的信息是断点的编号，我先把开始设的断点1号、2号删除了，再新增了两个断点——3号和4号，所以这时候再删除1号断点是不能成功的

对于后续的调试操作——进入函数、跳出函数和查看函数调用栈关系，我增加了add.c和max.c，再进行新的调试，所有的文件如下：

```
[TLDS@localhost text1021]$ ls
add.c  add.h  main.c  max.c  max.h
[TLDS@localhost text1021]$
```

add.h

```
int add( int a, int b);
~
```

max.h

```
int max( int a, int b);
~
```

add.c

```
int add( int a, int b)
{
  return a+b;
}
~
```

max.c

```c
int max( int a, int b)
{
  return a>b ? a:b;
}
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "add.h"
#include "max.h"

int main()
{
  int arr[6] = {0};
  int i=0;
  int a=0;
  int b=0;

  for( ; i<6; i++)
  {
    arr[i] = i;
  }

  a = add(3,5);
  b = max(3,5);

  exit(0);
}
```


首先，将这些程序进行编译、链接，有两种在程序中加入调试信息的方法。

方法一：在add.c、max.c的编译过程中加入调试信息

```
[TLDS@localhost text1021]$ gcc -c add.c -g
[TLDS@localhost text1021]$ gcc -c max.c -g
[TLDS@localhost text1021]$ gcc -o main main.c add.o max.o
[TLDS@localhost text1021]$ ./main
0
1
2
3
4
5
[TLDS@localhost text1021]$ gdb main
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-56.el6)
Copyright (C) 2010 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.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/TLDS/exercise/text1021/main...done.
(gdb) l
1       int add( int a, int b);
2
```


方法二：在三个程序的编译链接过程中加入调试信息

```
[TLDS@localhost text1021]$ gcc -o main main.c add.c max.c -g
[TLDS@localhost text1021]$ ./main
0
1
2
3
4
5
[TLDS@localhost text1021]$ gdb main
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-56.el6)
Copyright (C) 2010 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.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/TLDS/exercise/text1021/main...done.
(gdb) l
1       #include <stdio.h>
2       #include <stdlib.h>
3       #include "add.h"
4       #include "max.h"
```


开始调试：

在调用add()、max()处下断点

```
(gdb) b 15
Breakpoint 1 at 0x804843c: file main.c, line 15.
(gdb) b 19
Breakpoint 2 at 0x804846d: file main.c, line 19.
(gdb) b 20
Breakpoint 3 at 0x8048485: file main.c, line 20.
(gdb) info b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x0804843c in main at main.c:15
2       breakpoint     keep y   0x0804846d in main at main.c:19
3       breakpoint     keep y   0x08048485 in main at main.c:20
```

进入add()，并且查看函数调用栈关系

```
(gdb) c
Continuing.

Breakpoint 3, main () at main.c:20
20          b = max(3,5);   断点处停下
3: b = 0
2: a = 8
1: arr[i] = 6
(gdb) s
max (a=3, b=5) at max.c:5
5           return a>b ? a:b;
(gdb) bt
#0  max (a=3, b=5) at max.c:5
#1  0x08048499 in main () at main.c:20
```

退出add()，查看函数调用栈关系

```
(gdb) finish
Run till exit from #0  max (a=3, b=5) at max.c:5
0x08048499 in main () at main.c:20
20          b = max(3,5);
3: b = 0
2: a = 8
1: arr[i] = 6
Value returned is $1 = 5
(gdb) bt
#0  0x08048499 in main () at main.c:20
```

对比上面两张图，可以看到在main里看bt和在add中看bt的差别。同理，在max中能够看到 的main的调用信息，如果max中又调用了另一个函数，则 中查看bt的话，就能看到三个函数了。

以上，就是比较简单的gdb调试过程，如果想要了解更多关于gdb调试的知识，可以去看看《gdb完全用户手册》，谢谢您的观看。

想对作者说点什么

linux使用gdb调试程序完全教程 8161
转自 http://blog.csdn.net/gatieme 程序的调试过程主要有：单步执行，跳入函数，跳出函数，设置断点，设置观...

Linux GDB调试完全教程 1451
转自 http://blog.csdn.net/gatieme 本文将主要介绍linux下的强大调试工具是怎么完成这些工作的。 之所以要调试...

Linux环境下使用GDB调试C程序 6040
写这篇博客的目的是为了对gdb的常用命令做一个备忘，记录回顾下使用gdb的过程加深记忆。gdb是linux环境下...

Linux下gdb调试用法命令 9930
一直在Fedora平台下写opencv的程序，需要对程序进行调试，主要用的调试工具是gdb. gdb提供了如下功能： 1....

linux gdb调试基本命令 2453
Gdb调试 注意：在Gcc编译选项中一定要加入 –g gcc -g -o debug debug.c 1.启动调试 前置条件：编译生成执行码...

Linux gdb调试器用法全面解析 6.9万
GDB是GNU开源组织发布的一个强大的UNIX下的程序调试工具，GDB主要可帮助工程师完成下面4个方面的功...

用GDB调试程序（一） 57万
用GDB调试程序GDB概述−−−−GDB是GNU开源组织发布的一个强大的UNIX下的程序调试工具。或许，各位比...

下载 **linux下GDB调试教程** 08-06
这是中文版的，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，这给程序开发提供了极大...

linux下gdb调试基本命令学习 636
我们知道软件开发，调试技能是一个必备技能，学会调试就能快速高效地抓到bug，而gdb 是 GNU 调试器，Linu...

LINUX下GDB调试 4.6万
(注：本文实例在SecureCRT中得到验证,以下为全文转载：) 本文写给主要工作在Windows操作系统下而又需要开...

文章热词
linux 创建指定大小的文件　c读取每一行 linux　linux 跨网段ping不通　linux驱动程序开发前景如何　linux 打开opencv

相关热词
linux系统　linux系统表　linux系统号　linux系统卷　刷linux系统

博主推荐
fengbingchun 关注 729篇文章
Dablelv 关注 366篇文章
博文视点 关注 1786篇文章

帅气的羊羊杨 关注

原创	粉丝	喜欢	评论
27	11	5	5

等级：博客 2　访问：8647
积分：357　排名：24万+
勋章：恒

最新文章

浅谈智能指针

两个有序无头节点单链表的合并

C++—继承与多态

C++前序——(3)目标文件间的链接

C++前序——(2)目标文件

个人分类

数据结构	6篇
C语言	9篇
二进制	1篇
程序猿的自我修养	1篇
Linux	5篇

展开

归档

2018年6月	1篇
2018年5月	1篇
2018年4月	1篇
2017年12月	3篇
2017年11月	8篇

展开

热门文章

一个程序从源代码到可执行程序的过程
阅读量：3805

分页内存管理——虚拟地址到物理地址的转换
阅读量：1397

申请动态内存——malloc()函数及其扩展函数
阅读量：558

函数、全局变量、局部变量和动态内存的特点归纳
阅读量：245

Linux系统之下的基本gdb调试
阅读量：243

最新评论

分页内存管理——虚拟地址到物理地址...
qq_40780910：希望转载，可否授权？

链表——单链表
qq_39755395：[reply]weixin_41540461[/reply] 你好，因为链表使用完毕后，我们需要摧...

链表——单链表
weixin_41540461：摧毁链表不是应该把头结点也销毁吗？

函数、全局变量、局部变量和动态内存...
qq_39755395：[reply]YanyiRan0505[/reply] 嘿嘿

函数、全局变量、局部变量和动态内存...
YanyiRan0505：哇，写的不错，就是表格做的再精致一点就更好了、继续加油哦~

联系我们


官方公众号


区块链大本营