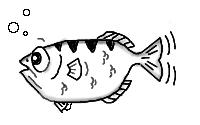
# [C编译: 使用gdb调试](http://www.cnblogs.com/vamei/archive/2013/04/03/2998652.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！



gdb是[the GNU Debugger](http://www.gnu.org/software/gdb/)的简称。它是一款UNIX平台的调试器(debugger)，可用于为C, C++, Objective-C, Java, Fortran等程序debug。

在gdb中，你可以通过设置断点(break point)来控制程序运行的进度，并查看断点时的变量和函数调用状况，从而发现可能的问题。在许多IDE中，gdb拥有图形化界面。

这里主要介绍gdb的命令行使用，并以C程序为例。测试使用的计算机是Mac OS系统。

### 启动gdb

下面的有两个C文件。(并没有bug。我们使用gdb来查看程序运行的细节)

程序的一个为test.c，其中有主程序main()。mean.c程序中定义了mean()函数，并在main()中调用。

test.c

[复制代码](javascript:void(0);)

#define ARRAYSIZE 4

float mean(float, float);

int main()

{

int i;

float a=4.5;

float b=5.5;

float rlt=0.0;

float array\_a[ARRAYSIZE]={1.0, 2.0, 3.0, 4.0};

float array\_b[ARRAYSIZE]={4.0, 3.0, 2.0, 1.0};

float array\_rlt[ARRAYSIZE];

for(i = 0; i < ARRAYSIZE - 1; i++) {

array\_rlt[i] = mean(array\_a[i], array\_b[i]);

}

rlt = mean(a, b);

return 0;

}

[复制代码](javascript:void(0);)

mean.c

float mean(float a, float b)

{

return (a + b)/2.0;

}

使用gcc同时编译上面两个程序。为了使用gdb对进行调试，必须使用-g选项(在编译时生成debugging信息):

$gcc -g -o test test.c mean.c

生成main可执行文件。

(如有必要，使用:

$chmod +x test

来增加用户的执行权限。)

进入gdb，准备调试程序:

$gdb test

进入gdb的互动命令行。

### 显示程序

我们可以直接显示某一行的程序，比如查看第9行程序:

(gdb) list 9

将显示以第9行为中心，总共10行的程序。我们实际上编译了两个文件，在没有说明的情况下，默认为主程序文件test.c:

[复制代码](javascript:void(0);)

4

5 int main()

6 {

7 int i;

8 float a=4.5;

9 float b=5.5;

10 float rlt=0.0;

11

12 float array\_a[ARRAYSIZE]={1.0, 2.0, 3.0, 4.0};

13 float array\_b[ARRAYSIZE]={4.0, 3.0, 2.0, 1.0};

[复制代码](javascript:void(0);)

如果要查看mean.c中的内容，需要说明文件名:

(gdb) list mean.c:1

可以具体说明所要列出的程序行的范围:

(gdb) list 5, 15

即显示5-15行的程序。

显示某个函数，比如:

(gdb) list mean

### 设置断点

我们可以运行程序:

(gdb) run

程序正常结束。

运行程序并没有什么有趣的地方。gdb的主要功能在于能让程序在中途暂停。

断点(break point)是程序执行中的一个位置。在gdb中，当程序运行到该位置时，程序会暂停，我们可以查看此时的程序状况，比如变量的值。

我们可以在程序的某一行设置断点，比如:

(gdb) break 16

将在test.c的第16行设置断点。

你可以查看自己设置的断点:

(gdb) info break

每个断点有一个识别序号。我们可以根据序号删除某个断点:

(gdb) delete 1

也可以删除所有断点:

(gdb) delete breakpoints

### 查看断点

设置断点，并使用run运行程序，程序将运行到16行时暂停。gdb显示:

Breakpoint 1, main () at test.c:16

16 for(i = 0; i < ARRAYSIZE - 1; i++) {

查看断点所在行:

(gdb) list

查看断点处的某个变量值:

(gdb) print a

(gdb) print array\_a

查看所有的局部变量:

(gdb) info local

查看此时的栈状态 (反映了函数调用，见[Linux从程序到进程](http://www.cnblogs.com/vamei/archive/2012/10/09/2715388.html)):

(gdb) info stack

可以更改变量的值。

(gdb) set var a=0.0

(gdb) set var array\_a={0.0, 0.0, 1.0, 1.0}

当程序继续运行时，将使用更改后的值。

如果我们将断点设置在:

(gdb) break mean.c:2

此时栈中有两个a，一个属于main()，一个属于mean()。我们可以用function::variable的方式区分:

(gdb) print mean::a

### 运行控制

让程序从断点开始，再多运行一行:

(gdb) step

也可以使用下面命令，从断点恢复运行，直到下一个断点:

(gdb) continue

使用run重新开始运行。

### 帮助

你可以通过gdb的帮助学到更多:

(gdb) help

或者更具体的命令:

(gdb) help info

### 退出

使用下面命令退出gdb:

(gdb) quit

### debug

下面是一个有bug的C程序，demo.c

[复制代码](javascript:void(0);)

#include <stdio.h>

struct node {

int element;

};

typedef struct node \*PNode;

int main()

{

int i;

PNode np=NULL;

for (i=0; i<10; i++) {

printf("Hello World!");

}

printf("%d \n", np->element);

}

[复制代码](javascript:void(0);)

程序可以成功编译，但运行将返回:

Segmentation fault

可以使用gdb来查找问题。

Updata:

经[水土精灵](http://www.weibo.com/u/1544903770)提醒: gdb命令可以只输入首字母的缩写，比如c代表continue，b代表break，q代表quit。