GDB 调试

首先有这样一个文件：：

test.c (~/LinuxCode/基础部分/GDB/args) - VIM

```
 1 #include <stdio.h>
 2
 3 int main(int argc, const char* argv[])
 4 {
 5     printf("args num = %d\n", argc);
 6     for(int i=0; i<argc; ++i)
 7     {
 8         printf("arg%d: %s\n", i, argv[i]);
 9     }
10     return 0;
11 }
```

现在编译这个文件，加入调试信息-g：gcc test.c -g -o app
执行完这句后，执行命令 gdb app:

```
jack@jack-Ubuntu:~/LinuxCode/基础部分/GDB/args$ gdb app
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 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 "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from app...done.
(gdb)
```

进入 gdb 调试状态，等待参数的输入：

```
(gdb) set args sss xxx 444
(gdb) r
Starting program: /home/jack/LinuxCode/基础部分/GDB/args/app sss xxx 444
args num = 4
arg0: /home/jack/LinuxCode/基础部分/GDB/args/app
arg1: sss
arg2: xxx
arg3: 444
[Inferior 1 (process 6483) exited normally]
(gdb)
```

由于没有打断点，所以执行到此处程序全部运行完毕！

接下来以一个具体的例子学习 gdb 的调试方法！
首先有这么几个文件：

```c
#include <stdio.h>
#include "sort.h"

void main()
{
    int i;
    //定义整型数组
    int array[] = { 12, 5, 33, 6, 10, 35, 67, 89, 87, 65, 54, 24, 58, 92, 100, 24, 46, 78, 99, 200, 55, 44, 33, 22, 11, 71, 2, 4, 86, 8, 9 };
    int array2[] = { 12, 5, 33, 6, 10, 35, 67, 89, 87, 65, 54, 24, 58, 92, 100, 24, 46, 78, 99, 200, 55, 44, 33, 22, 11, 71, 2, 4, 86, 8, 9 };
    //计算数组长度
    int len = sizeof(array) / sizeof(int);
    //遍历数组
    printf("Sort Array:\n");
    for (i = 0; i < len; ++i)
    {
        printf("%d\t", array[i]);
    }
    printf("\n");
    // selectionSort
    selectionSort(array, len);
    // printf
    printf("Selection Sort:\n");
    for (i = 0; i < len; ++i)
    {
        printf("%d  ", array[i]);
    }
    // insertionSort
    insertionSort(array2, len);
    // printf
    printf("\n==========Gorgeous Split Line==========\n");
    printf("Insertion Sort: \n");
    for (i = 0; i < len; ++i)
    {
        printf("%d  ", array2[i]);
    }
    printf("\n");
}
```

```c
/*-------------------- 排序规则 --------------------

        它的工作原理是每一次从待排序的数据元素中选出最小（或最大）的一个元素，存放在序列的起始位置，直到全部待排序的数据元素排完。

        稳定性:选择排序是不稳定的排序方法 如:[5,5,3]

-------------------------------------------------*/

//选择排序(升序排列)
void selectionSort(int *array, int len)
{
    int min = 0;    // 指向最小的元素的位置
    // 外层循环
    for (int i = 0; i < len - 1; ++i)
    {
        min = i;
        // 内存循环
        for (int j = i + 1; j < len; ++j)
        {
            // 判断
            if (array[min] > array[j])
            {
                // 保存最小的元素的位置
                min = j;
            }
        }
        // 判断是否需要交换
        if (min != i)
        {
            // 找到了新的最小值
            // 交换
            int tmp = array[min];
            array[min] = array[i];
            array[i] = tmp;
        }
    }
}
```

```c
#include <stdio.h>
#include <stdlib.h>
/****************** 排序规则 ******************

    每次处理就是将无序数列的第一个元素与有序数列的元素从后往前逐个进行比较，找出插入位置，将该元素插入到有序数列的合适位置中。

    稳定性:插入排序是稳定的
*********************************************/
//插入排序算法(升序排列)
void insertionSort(int *array, int len)
{
    int tmp = 0;    // 存储基准数
    int index = 0;  // 坑的位置
    // 遍历无序序列
    for (int i = 1; i < len; ++i)
    {
        index = i;
        tmp = array[i];
        // 遍历有序序列(从后往前)
        for (int j = i - 1; j >= 0; --j)
        {
            // 基准数根有序序列中的元素比较
            if (tmp < array[j])
            {
                // 有序序列元素后移
                array[j + 1] = array[j];
                // 坑的位置
                index = j;
            }
            else
            {
                break;
            }
        }
        // 填坑
        array[index] = tmp;
    }
```

现在开始调试：

```
jack@jack-Ubuntu:~/LinuxCode/基础部分/GDB$ gcc *.c -g -o app -std=c99
jack@jack-Ubuntu:~/LinuxCode/基础部分/GDB$ ls
app  args  insert_sort.c  main.c  makefile  select_sort.c  sort.h
jack@jack-Ubuntu:~/LinuxCode/基础部分/GDB$ gdb app
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 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 "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from app...done.
(gdb)
```

现在敲入 'l'，将显示被调试文件的内容：

```
(gdb) l
1       #include <stdio.h>
2       #include "sort.h"
3
4       void main()
5       {
6               int i;
7               //◆◆◆◆◆◆◆◆◆◆◆
8               int array[] = { 12, 5, 33, 6, 10, 35, 67, 89, 87, 65, 54, 24, 58, 92, 100, 24, 46, 78, 99, 200, 55, 44, 33, 22, 11, 71, 2, 4, 86, 8, 9
};
9               int array2[] = { 12, 5, 33, 6, 10, 35, 67, 89, 87, 65, 54, 24, 58, 92, 100, 24, 46, 78, 99, 200, 55, 44, 33, 22, 11, 71, 2, 4, 86, 8, 9
 };
10
(gdb)
```

接下来输入：show listsize

```
(gdb) show listsize
Number of source lines gdb will list by default is 10.
(gdb)
```

可见，gdb 默认显示文件内容为 10 行！

接下来输入:set listsize 20

```
(gdb) set listsize 20
(gdb) l
11              //◆◆◆◆◆◆◆◆鳖◆
12              int len = sizeof(array) / sizeof(int);
13              //◆◆◆◆◆◆◆◆
14              printf("Sort Array:\n");
15              for (i = 0; i < len; ++i)
16              {
17                      printf("%d\t", array[i]);
18              }
19              printf("\n");
20
21              // selectionSort
22              selectionSort(array, len);
23              // printf
24              printf("Selection Sort:\n");
25              for (i = 0; i < len; ++i)
26              {
27                      printf("%d  ", array[i]);
28              }
29
30              // insertionSort
(gdb)
```

于是显示文件下面的 20 行内容。

接下来输入：l 4

```
(gdb) set listsize 10
(gdb) l 4
1       #include <stdio.h>
2       #include "sort.h"
3
4       void main()
5       {
6               int i;
7               //◆◆◆◆◆◆◆◆◆◆◆
8               int array[] = { 12, 5, 33, 6, 10, 35, 67, 89, 87, 65, 54, 24, 58, 92, 100, 24, 46, 78, 99, 200, 55, 44, 33, 22, 11, 71, 2, 4, 86, 8, 9
};
9               int array2[] = { 12, 5, 33, 6, 10, 35, 67, 89, 87, 65, 54, 24, 58, 92, 100, 24, 46, 78, 99, 200, 55, 44, 33, 22, 11, 71, 2, 4, 86, 8, 9
 };
10
(gdb)
```

将显示第四行的上下文。
如果需要查看其他文件，则：l insert_sort.c:15

```
(gdb) l insert_sort.c:15
10              ◆ţ◆◆◆:◆◆◆◆◆◆◆◆◆◆ţ◆◆
11
12      ********************************************/
13
14      //◆◆◆◆◆◆◆◆◆◆燁(◆◆◆◆◆◆◆◆◆)
15      void insertionSort(int *array, int len)
16      {
17              int tmp = 0;    // ◆浒◆◆
18              int index = 0;  // ◆q◆λ◆
19              // ◆◆◆◆◆◆◆◆◆◆◆
(gdb)
```

将显示 insert_sort.c 第 15 行上下文内容。
继续输入：l insert_sort.c:insertionSort

```
(gdb) l insert_sort.c:insertionSort
11
12      ********************************************/
13
14      //◆◆◆◆◆◆◆◆◆◆燁(◆◆◆◆◆◆◆◆◆)
15      void insertionSort(int *array, int len)
16      {
17              int tmp = 0;    // ◆浒◆◆
18              int index = 0;  // ◆q◆λ◆
19              // ◆◆◆◆◆◆◆◆◆◆◆
20              for (int i = 1; i < len; ++i)
(gdb)
```

将显示函数名处上下文内容。
接下来设置断点，输入：b 行号

```
(gdb) b 12
Breakpoint 1 at 0x4008b0: file main.c, line 12.
(gdb) b14
Undefined command: "b14".  Try "help".
(gdb) b 14
Breakpoint 2 at 0x4008ba: file main.c, line 14.
(gdb) b 15
Breakpoint 3 at 0x4008c4: file main.c, line 15.
(gdb) b 17
Breakpoint 4 at 0x4008d0: file main.c, line 17.
(gdb) b 19
Breakpoint 5 at 0x400905: file main.c, line 19.
(gdb) b 24
Breakpoint 6 at 0x400926: file main.c, line 24.
(gdb) b 27
Breakpoint 7 at 0x40093c: file main.c, line 27.
(gdb)
```

已经设置多个断点。
输入 i(info) b(break)可以查看设置的断点：

```
(gdb) i b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000004008b0 in main at main.c:12
2       breakpoint     keep y   0x00000000004008ba in main at main.c:14
3       breakpoint     keep y   0x00000000004008c4 in main at main.c:15
4       breakpoint     keep y   0x00000000004008d0 in main at main.c:17
5       breakpoint     keep y   0x0000000000400905 in main at main.c:19
6       breakpoint     keep y   0x0000000000400926 in main at main.c:24
7       breakpoint     keep y   0x000000000040093c in main at main.c:27
(gdb)
```

接下来用 d(del)命令删除断点：

```
(gdb) d 1
(gdb) l b
Function "b" not defined.
(gdb) i b
Num     Type           Disp Enb Address            What
2       breakpoint     keep y   0x00000000004008ba in main at main.c:14
3       breakpoint     keep y   0x00000000004008c4 in main at main.c:15
4       breakpoint     keep y   0x00000000004008d0 in main at main.c:17
5       breakpoint     keep y   0x0000000000400905 in main at main.c:19
6       breakpoint     keep y   0x0000000000400926 in main at main.c:24
7       breakpoint     keep y   0x000000000040093c in main at main.c:27
(gdb) d 2 3
(gdb) i b
Num     Type           Disp Enb Address            What
4       breakpoint     keep y   0x00000000004008d0 in main at main.c:17
5       breakpoint     keep y   0x0000000000400905 in main at main.c:19
6       breakpoint     keep y   0x0000000000400926 in main at main.c:24
7       breakpoint     keep y   0x000000000040093c in main at main.c:27
(gdb)
```

接下来用 dis 命令无效化断点：

```
(gdb) dis 4 5
(gdb) i b
Num     Type           Disp Enb Address            What
4       breakpoint     keep n   0x00000000004008d0 in main at main.c:17
5       breakpoint     keep n   0x0000000000400905 in main at main.c:19
6       breakpoint     keep y   0x0000000000400926 in main at main.c:24
7       breakpoint     keep y   0x000000000040093c in main at main.c:27
(gdb)
```

用 ena 命令有效化断点：

```
(gdb) ena 4 5
(gdb) ib
Undefined command: "ib".  Try "help".
(gdb) i b
Num     Type           Disp Enb Address            What
4       breakpoint     keep y   0x00000000004008d0 in main at main.c:17
5       breakpoint     keep y   0x0000000000400905 in main at main.c:19
6       breakpoint     keep y   0x0000000000400926 in main at main.c:24
7       breakpoint     keep y   0x000000000040093c in main at main.c:27
(gdb)
```

运行调试程序：

首先用 r 命令，接着 p i 命令查看 i 的当前值：

```
(gdb) b 17
Breakpoint 10 at 0x4008d0: file main.c, line 17.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/jack/LinuxCode/基础部分/GDB/app
Sort Array:

Breakpoint 10, main () at main.c:17
17                      printf("%d\t", array[i]);
(gdb) p i
$2 = 0
(gdb)
```

当然也可以条件设置： b 17 if i == 10

```
(gdb) l
12              int len = sizeof(array) / sizeof(int);
13              //◆◆◆◆◆◆◆
14              printf("Sort Array:\n");
15              for (i = 0; i < len; ++i)
16              {
17                      printf("%d\t", array[i]);
18              }
19              printf("\n");
20
21              // selectionSort
(gdb) b 17 if i == 10
Breakpoint 11 at 0x4008d0: file main.c, line 17.
(gdb) c
Continuing.

Breakpoint 11, main () at main.c:17
17                      printf("%d\t", array[i]);
(gdb) p i
$3 = 10
(gdb)
```

使用 ptype 查看变量类型：

```
(gdb) ptype i
type = int
(gdb) ptype array
type = int [31]
(gdb)
```

使用 n 和 display 命令单步执行程序并查看变量内容：

```
(gdb) n
15              for (i = 0; i < len; ++i)
(gdb) display i
1: i = 0
(gdb) n

Breakpoint 1, main () at main.c:17
17                      printf("%d\t", array[i]);
1: i = 1
(gdb) display i
2: i = 1
(gdb) n
15              for (i = 0; i < len; ++i)
2: i = 1
1: i = 1
(gdb) n

Breakpoint 1, main () at main.c:17
17                      printf("%d\t", array[i]);
2: i = 2
1: i = 2
(gdb) display array[i]
3: array[i] = 33
(gdb)
```

用 i display 查看打印变量：

```
(gdb) i display
Auto-display expressions now in effect:
Num Enb Expression
3:   y  array[i]
2:   y  i
1:   y  i
```

用 undisplay 取消打印:

```
(gdb) undisplay 1
(gdb) undisplay 2
(gdb) n
15              for (i = 0; i < len; ++i)
3: array[i] = 35
(gdb) i display
Auto-display expressions now in effect:
Num Enb Expression
3:   y  array[i]
(gdb)
```

可以看到，i 没有再打印了！
使用 c 命令 continue 下去：

```
(gdb) c
Continuing.

Breakpoint 1, main () at main.c:17
17                      printf("%d\t", array[i]);
3: array[i] = 87
(gdb)
Continuing.

Breakpoint 1, main () at main.c:17
17                      printf("%d\t", array[i]);
3: array[i] = 65
(gdb)
```

用 s(step)调到函数内部，finish 跳转回去：

```
(gdb) s

Breakpoint 1, main () at main.c:17
17                       printf("%d\t", array[i]);
3: array[i] = 54
(gdb) s
__printf (format=0x400b64 "%d\t") at printf.c:28
28      printf.c: 没有那个文件或目录.
(gdb) finish
Run till exit from #0  __printf (format=0x400b64 "%d\t") at printf.c:28
main () at main.c:15
15              for (i = 0; i < len; ++i)
3: array[i] = 54
Value returned is $1 = 3
(gdb)
```