**段错误**

**所谓的段错误就是指访问的内存超出了系统所给这个程序的内存空间**，通常这个值是由gd tr来保存的，他是一个48位的寄存器，其中的32位是保存由它指向的 gdt表，后13位保存 相应于gdt的下标，最后3位包括了程序是否在内存中以及程序的在cpu中的运行级别，指向 的gdt是由以64位为一个单位的表，在这张表中就保存着程序运行的代码段以及数据段的起 始地址以及与此相应的段限和页面交换还有程序运行级别还有内存粒度等等的信息。

几种典型的段错误1，

int main(void){[1]

char\*s ="hello world";

\*s ='H';

}

被装载时，系统把“hello world” 连同其它的字符串和const型数据放入到内存的只读区。执行时，一个变量s被设为指向该字符串的位置，当再试图向该位置写时，就会产生段错误。

2，

int\*ptr = NULL;

\*ptr =1;

因为该代码只创建了一个空指针，并没有指向一个具体空间，当赋值时，产生段错误。

3，

int main(void){

main();

return0;

}

无限递归，这会导致栈溢出，也会产生段错误。

### 利用gdb逐步查找段错误

这种方法也是被大众所熟知并广泛采用的方法，首先我们需要一个带有调试信息的可执行程序，所以我们加上“-g -rdynamic"的参数进行编译，然后用[gdb](http://baike.baidu.com/view/639266.htm)调试运行这个新编译的程序,具体步骤如下:

|  |
| --- |
| xiaosuo@gentux test $ gcc -g -rdynamic d.c  xiaosuo@gentux test $ gdb ./a.out  GNU gdb 6.5  Copyright (C) 2006 Free Software Foundation, Inc.  GDB is free software, covered by the GNU General Public License, and you are  welcome to change it and/or distribute copies of it under certain conditions.  Type "show copying" to see the conditions.  There is absolutely no warranty for GDB. Type "show warranty" for details.  This GDB was configured as "i686-pc-linux-gnu"...Using host libthread\_db library "/lib/libthread  (gdb) r  Starting program: /home/xiaosuo/test/a.out  Program received signal SIGSEGV, Segmentation fault.  0x08048524 in dummy\_function () at d.c:4  4 \*ptr = 0x00;  (gdb) |

不用一步步调试我们就找到了出错位置d.c文件的第4行，其实就是如此的简单。