前幾天領導安排一個小項目，大意是解決這樣一個問題：

在Linux系統下，進程可能由於各種原因崩潰，此時我們要找到出問題的源代碼在某一個文件的具體行號，這樣調試起來就會方便，高效很多，可能是公司項目要用到，想想挺有意思的，加上自己本身是個Linux狂熱者，最終花了兩三天解決了這個問題，當然我的領導我們稱之為專家指點了我很多，廢話少說，下面是解決問題的思路和步驟以及自己的一些想法

解決該問題的大體思路是這樣的：在Linux下，進程崩潰時內核（也就是我們所謂的操作系統）會向進程發送信號，比如我們程序運行崩潰時經常會看到segmentation falt這樣的信息，這是進程非法操作內存，內核會向進程發送SIGENV信號，那麼我們憑什麼可以找到進程崩潰的原因對應的源代碼的位置呢，我們知道， 每個進程都有自己的堆棧，當某個進程崩潰時，堆棧裡保存了一些關鍵信息，通過這些信息我們可以定位到出錯的源代碼位置，那麼我們怎麼來獲得進程崩潰時堆棧的信息呢，請記住，Linux是當今世界上最為強大的操作系統(不管你信不信，反正我是信了，^_^),在Linux系統裡有個backtrace這些個函數可以獲得當前堆棧信息，好了，到這裡問題已經解決了一半，backtrace信息中的有一個地址包含了出錯代碼在文件中的偏移量

需要注意的是編譯時一定要加上-g和-rdynamic參數

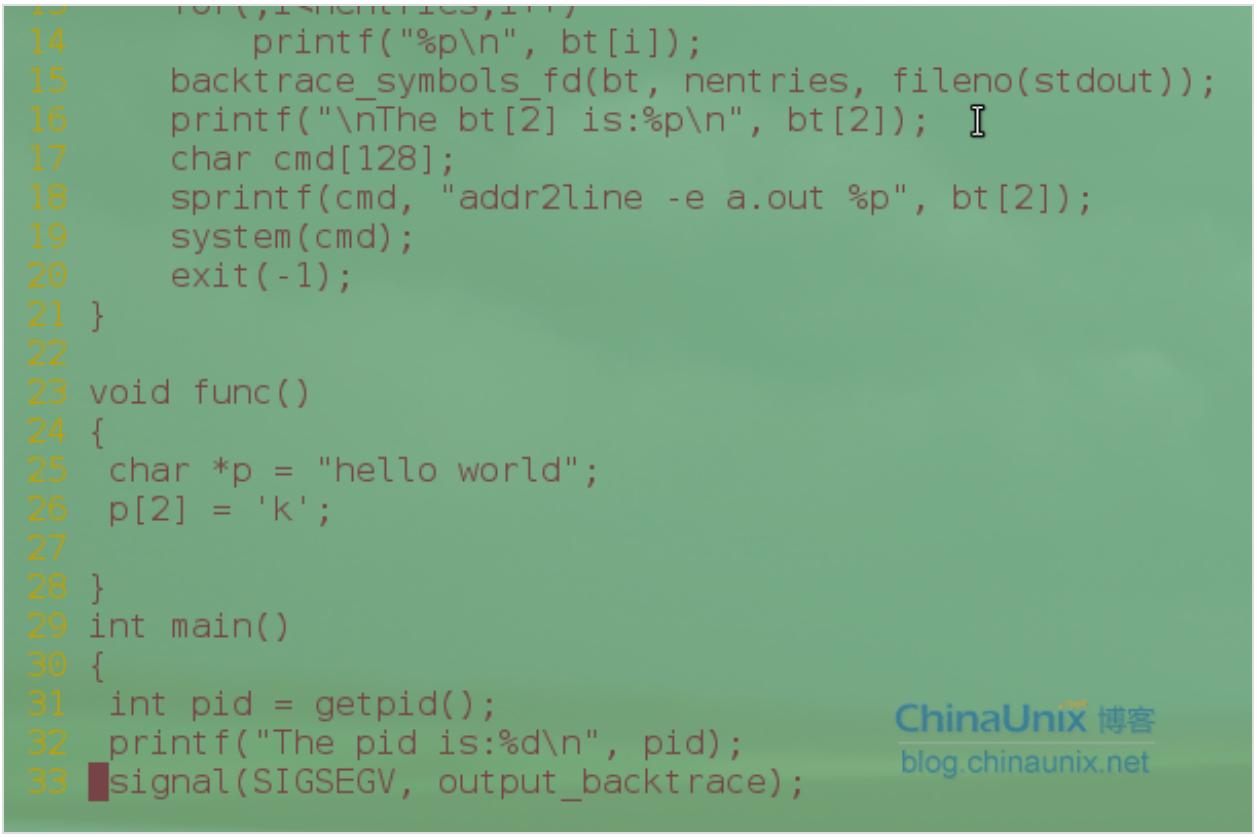
如果我們使用的是靜態庫或者出錯的代碼不在動態庫中，那麼我們可以直接用命令"addr2line -e 可執行文件名偏移地址"打印出出錯的代碼行，下面是具體步驟

在測試程序的29行非法操作了內存

```
14          printf("%p\n", bt[i]);
15      backtrace_symbols_fd(bt, nentries, fileno(stdout));
16      printf("\nThe bt[2] is:%p\n", bt[2]);
17      char cmd[128];
18      sprintf(cmd, "addr2line -e a.out %p", bt[2]);
19      system(cmd);
20      exit(-1);
21  }
22
23  void func()
24  {
25   char *p = "hello world";
26   p[2] = 'k';
27
28  }
29  int main()
30  {
31   int pid = getpid();
32   printf("The pid is:%d\n", pid);
33  signal(SIGSEGV, output_backtrace);
```


我們把addr2line**命令放在程序裡面做了，在代碼中也可以看得到，下面這幅圖片是程序的輸出，可以看到**

打印出源代碼出錯的行數為29



如果crash在一個動態庫so裡面，比較麻煩一點，此時addr2line不能直接給出代碼行。因為我們都知道，so裡面的地址在可執行文件裝載的時候， 是可以被reallocate的。所以，如果只有一個so的地址，要找出對應代碼行的話，送給addr2line的參數地址就是一個偏移地址，這裡的偏移地址就是backtrace中的地址減去動態庫加載的時候的基地址，這個基地址我們可以通過/proc/pid/maps這個文件找到，pid是當前進程號，下面是具體步驟，跟之前的步驟類似，只不過我們為了測試，將func函數編譯進了一個動態庫裡面

這裡我們只給出測試文件，具體怎麼實現動態庫，這個很容易，不在此多提，下面是測試程序的一部分

```
    int nentries = backtrace(bt, sizeof(bt)/sizeof(bt[0]));
    backtrace_symbols_fd(bt, nentries, backtrace_file);
    //backtrace_symbols_fd(bt, nentries, fileno(stdout));
    char*So_Name = FindSoName();
    char*BaseAddress = FindBaseAddress(So_Name);
    printf("The Stack Address is:%p\n", bt[2]);
    char Stack[50];
    sprintf(Stack, "%p", bt[2]);
    int OffsetAddress = strtol(Stack, NULL, 16) - strtol(Ba
;
    printf("OffsetAddress:%d\n", OffsetAddress);
    PrintLineNumber(So_Name, OffsetAddress);
    exit(-1);
}

int main()
{
 int pid = getpid();
 printf("The pid is:%d\n", pid);
 signal(SIGSEGV, OutputBacktrace);
 func();
 return 0;
}
```


我們在`/proc/pid/maps`**文件中找出出錯動態庫加載的基地址，用**`backtrace`**中的地址與基地址相減得到偏移地址就可以了，下面是程序輸出**

```
root@debian6-dev:/home/mcir/kevin/C++/ErrorLocation/so# ./a.out
The pid is:8903
The .so name is:libmyhello.so
The Base address of the libmyhello.so is:7f8daa25e000
The Stack Address is:0x7f8daa25e5d7
OffsetAddress:1495
/home/mcir/kevin/C++/ErrorLocation/so/hello.c:5
root@debian6-dev:/home/mcir/kevin/C++/ErrorLocation/so#
```

可以看到，我們用hello.c做的動態庫，定位到代碼出錯在hello.c的第五行