# 使用c++编写和使用.so动态链接库

<https://blog.51cto.com/quietmadman/1394968>

gcc-fPIC -shared -o mylib.so mylib.c

这里的共享对象其实全称是动态共享对象文件（Dynamic Shared Objects，简写为DSO）

-fPIC：地址无关代码（Position-Independent Code），该技术主要用于解决SO中对绝对地址的重定位问题；

链接so的时候是如下这样

g++ -o use\_mylib use\_mylib.cpp ./mylib.so

# 在linux程序中输出函数的调用栈

<http://hutaow.com/blog/2013/10/11/dump-stack-with-backtrace/>

编译时需要加上-rdynamic参数，以得到符号名称，像下面这样：

<https://blog.csdn.net/astrotycoon/article/details/8142588>

我们还可以利用这backtrace来定位段错误位置。

通常情况系，程序发生段错误时系统会发送SIGSEGV信号给程序，缺省处理是退出函数。我们可以使用 signal(SIGSEGV, &your\_function);函数来接管SIGSEGV信号的处理，程序在发生段错误后，自动调用我们准备好的函数，从而在那个函数里来获取当前函数调用栈。

#include <stdio.h>

#include <stdlib.h>

#include <stddef.h>

#include <execinfo.h>

#include <signal.h>

void dump(int signo)

{

void \*buffer[30] = {0};

size\_t size;

char \*\*strings = NULL;

size\_t i = 0;

size = backtrace(buffer, 30);

fprintf(stdout, "Obtained %zd stack frames.nm\n", size);

strings = backtrace\_symbols(buffer, size);

if (strings == NULL)

{

perror("backtrace\_symbols.");

exit(EXIT\_FAILURE);

}

for (i = 0; i < size; i++)

{

fprintf(stdout, "%s\n", strings[i]);

}

free(strings);

strings = NULL;

exit(0);

}

void func\_c()

{

\*((volatile char \*)0x0) = 0x9999;

}

void func\_b()

{

func\_c();

}

void func\_a()

{

func\_b();

}

int main(int argc, const char \*argv[])

{

if (signal(SIGSEGV, dump) == SIG\_ERR)

perror("can't catch SIGSEGV");

func\_a();

return 0;

}

---------------------

作者：astrotycoon

来源：CSDN

原文：https://blog.csdn.net/astrotycoon/article/details/8142588

版权声明：本文为博主原创文章，转载请附上博文链接！

# 使用linux下载软件

wget http://www.minjieren.com/wordpress-3.1-zh\_CN.zip

<https://www.cnblogs.com/peida/archive/2013/03/18/2965369.html>

# gdb调试(no debugging symbols found)

编译时未加调试选项（(g)cc/g++  -g ... ）

https://blog.csdn.net/ixidof/article/details/4791968

# The program is not being run

这是gdb在提示程序还没有运行，需要自己去执行run来运行这个程序

<https://lellansin.wordpress.com/2013/04/18/gdb-%E5%9F%BA%E6%9C%AC%E7%94%A8%E6%B3%95/>