

C++ 信号处理

信号是由操作系统传给进程的中断，会提早终止一个程序。在 UNIX、LINUX、Mac OS X 或 Windows 系统上，可以通过按 Ctrl+C 产生中断。

有些信号不能被程序捕获，但是下表所列信号可以在程序中捕获，并可以基于信号采取适当的动作。这些信号是定义在 C++ 头文件 <csignal> 中。

| 信号 | 描述 |
|---------|----------------------------|
| SIGABRT | 程序的异常终止，如调用 abort 。 |
| SIGFPE | 错误的算术运算，比如除以零或导致溢出的操作。 |
| SIGILL | 检测非法指令。 |
| SIGINT | 接收到交互注意信号。 |
| SIGSEGV | 非法访问内存。 |
| SIGTERM | 发送到程序的终止请求。 |

signal() 函数

C++ 信号处理库提供了 **signal** 函数，用来捕获突发事件。以下是 signal() 函数的语法：

```
void (*signal (int sig, void (*func)(int)))(int);
```

这个函数接收两个参数：第一个参数是一个整数，代表了信号的编号；第二个参数是一个指向信号处理函数的指针。

让我们编写一个简单的 C++ 程序，使用 signal() 函数捕获 SIGINT 信号。不管您想在程序中捕获什么信号，您都必须使用 **signal** 函数来注册信号，并将其与信号处理程序相关联。看看下面的实例：

实例

```
#include <iostream>
#include <csignal>
#include <unistd.h>

using namespace std;

void signalHandler( int signum )
{
```



C++ 接口 (抽象类)

C++ 高级教程

C++ 文件和流

C++ 异常处理

C++ 动态内存

C++ 命名空间

C++ 模板

C++ 预处理器

C++ 信号处理

C++ 多线程

C++ Web 编程

C++ 资源库

C++ STL 教程

C++ 标准库

C++ 有用的资源

C++ 实例

```
cout << "Interrupt signal (" << signalnum << ") received.\n";

// 清理并关闭
// 终止程序

exit(signalnum);

}

int main ()
{
    // 注册信号 SIGINT 和信号处理程序
    signal(SIGINT, signalHandler);

    while(1){
        cout << "Going to sleep...." << endl;
        sleep(1);
    }

    return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Going to sleep....
Going to sleep....
Going to sleep....
```

现在，按 Ctrl+C 来中断程序，您会看到程序捕获信号，程序打印如下内容并退出：

```
Going to sleep....
Going to sleep....
Going to sleep....
Interrupt signal (2) received.
```

raise() 函数

您可以使用函数 **raise()** 生成信号，该函数带有一个整数信号编号作为参数，语法如下：

```
int raise (signal sig);
```

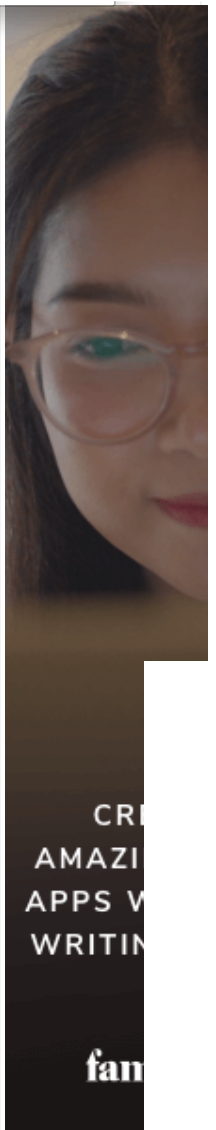
在这里，**sig** 是要发送的信号编号，这些信号包括：SIGINT、SIGABRT、SIGFPE、SIGILL、SIGSEGV、SIGTERM、SIGHUP。以下是我们使用 **raise()** 函数内部生成信号的实例：

实例

```
#include <iostream>
#include <csignal>
#include <unistd.h>

using namespace std;

void signalHandler( int signalum )
{
```



反馈/建议

```

    cout << "Interrupt signal (" << signum << ") received.\n";

    // 清理并关闭
    // 终止程序

    exit(signum);
}

int main ()
{
    int i = 0;
    // 注册信号 SIGINT 和信号处理程序
    signal(SIGINT, signalHandler);

    while(++i){
        cout << "Going to sleep...." << endl;
        if( i == 3 ){
            raise( SIGINT);
        }
        sleep(1);
    }

    return 0;
}

```

当上面的代码被编译和执行时，它会产生下列结果，并会自动退出：

```

Going to sleep....
Going to sleep....
Going to sleep....
Interrupt signal (2) received.

```

← C++ 预处理器

C++ 多线程 →



2 篇笔记



写笔记



Sleep 函数

功能：执行挂起一段时间，也就是等待一段时间在继续执行

用法：Sleep(时间)

3 注意：

- （1）Sleep是区分大小写的，有的编译器是大写，有的是小写。
- （2）Sleep括号里的时间，在windows下是已毫秒为单位，而Linux是已秒为单位。

```

#include <iostream>
#include <windows.h>

using namespace std;

int main()
{

```



反馈/建议

```

int a = 1;
while (a)
{
    cout << "欢迎来到菜鸟教程!" << endl;
    Sleep(100);
}
system("pause");
return 0;
}

```

爱撸码的张小白 2年前 (2017-10-11)



Linux 用 `#include <unistd.h>` 和 `sleep()`, Windos 用 `#include <windows.h>` 和 `Sleep()`。

3

```

#include <iostream>
#include <csignal>
#include <windows.h>

using namespace std;

void signalHandler(int signum)
{
    cout << "Interrupt signal (" << signum << ") received.\n";

    // 清理并关闭
    // 终止程序

    exit(signum);
}

int main()
{
    int i = 0;
    // 注册信号 SIGINT 和信号处理程序
    signal(SIGINT, signalHandler);

    while (++i) {
        cout << "Going to sleep...." << endl;
        if (i == 3) {
            raise(SIGINT);
        }
        Sleep(1);
    }

    return 0;
}

```

冰封绝杀 3个月前 (08-03)



反馈/建议

在线实例

- [HTML 实例](#)
- [CSS 实例](#)
- [JavaScript 实例](#)
- [Ajax 实例](#)
- [jQuery 实例](#)
- [XML 实例](#)
- [Java 实例](#)

字符集&工具

- [HTML 字符集设置](#)
- [HTML ASCII 字符集](#)
- [HTML ISO-8859-1](#)
- [HTML 实体符号](#)
- [HTML 拾色器](#)
- [JSON 格式化工具](#)

最新更新

- [Python redis 使...](#)
- [Windows10 MYSQ...](#)
- [Docker 镜像加速](#)
- [Debian Docker 安装](#)
- [C 库函数 -...](#)
- [Linux groupadd ...](#)
- [CSS var\(\) 函数](#)

站点信息

- [意见反馈](#)
- [合作联系](#)
- [免责声明](#)
- [关于我们](#)
- [文章归档](#)

关注微信



Copyright © 2013-2019 **菜鸟教程**
runoob.com All Rights Reserved.
备案号: 闽ICP备15012807号-1



反馈/建议