## 1、信号的基本概念

信号（signal）是软件中断，是进程之间相互传递消息的一种方法，用于通知进程发生了事件，但是，不能给进程传递任何数据。

信号产生的原因有很多，在Linux下，可以用kill和killall命令发送信号。

## 2、信号的类型

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 信号值 | 默认处理动作 | 发出信号的原因 |
| SIGHUP | 1 | A | 终端挂起或者控制进程终止 |
| **SIGINT** | **2** | **A** | **键盘中断Ctrl+c** |
| SIGQUIT | 3 | C | 键盘的退出键被按下 |
| SIGILL | 4 | C | 非法指令 |
| SIGABRT | 6 | C | 由abort(3)发出的退出指令 |
| SIGFPE | 8 | C | 浮点异常 |
| **SIGKILL** | **9** | **AEF** | **采用kill -9 进程编号 强制杀死程序。** |
| SIGSEGV | 11 | C | 无效的内存引用 |
| SIGPIPE | 13 | A | 管道破裂，写一个没有读端口的管道。 |
| **SIGALRM** | **14** | **A** | **由alarm(2)发出的信号** |
| **SIGTERM** | **15** | **A** | **采用“kill 进程编号”或“killall 程序名”通知程序。** |
| **SIGUSR1** | **10** | **A** | **用户自定义信号1** |
| **SIGUSR2** | **12** | **A** | **用户自定义信号2** |
| **SIGCHLD** | **17** | **B** | **子进程结束信号** |
| SIGCONT | 18 |  | 进程继续（曾被停止的进程） |
| SIGSTOP | 19 | DEF | 终止进程 |
| SIGTSTP | 20 | D | 控制终端（tty）上按下停止键 |
| SIGTTIN | 21 | D | 后台进程企图从控制终端读 |
| SIGTTOU | 22 | D | 后台进程企图从控制终端写 |

处理动作一项中的字母含义如下

A 缺省的动作是终止进程。

B 缺省的动作是忽略此信号，将该信号丢弃，不做处理。

C 缺省的动作是终止进程并进行内核映像转储（core dump），内核映像转储是指将进程数据在内存的映像和进程在内核结构中的部分内容以一定格式转储到文件系统，并且进程退出执行，这样做的好处是为程序员 提供了方便，使得他们可以得到进程当时执行时的数据值，允许他们确定转储的原因，并且可以调试他们的程序。

D 缺省的动作是停止进程，进入停止状态的程序还能重新继续，一般是在调试的过程中。

E 信号不能被捕获。

F 信号不能被忽略。

## 3、信号的处理

进程对信号的处理方法有三种：

1）对该信号的处理采用系统的默认操作，大部分的信号的默认操作是终止进程。

2）设置中断的处理函数，收到信号后，由该函数来处理。

3）忽略某个信号，对该信号不做任何处理，就像未发生过一样。

signal函数可以设置程序对信号的处理方式。

函数声明：

sighandler\_t signal(int signum, sighandler\_t handler);

参数signum表示信号的编号。

参数handler表示信号的处理方式，有三种情况：

1）SIG\_DFL：恢复参数signum所指信号的处理方法为默认值。

2）一个自定义的处理信号的函数，信号的编号为这个自定义函数的参数。

3）SIG\_IGN：忽略参数signum所指的信号。

## 4、信号有什么用

服务程序运行在后台，如果想让中止它，杀掉不是个好办法，因为程序被杀的时候，程序突然死亡，没有安排善后工作。

如果向服务程序发送一个信号，服务程序收到这个信号后，调用一个函数，在函数中编写善后的代码，程序就可以有计划的退出。

向服务程序发送0的信号，可以检测程序是否存活。

## 5、信号应用示例

在实际开发中，在main函数开始的位置，程序员会先屏蔽掉全部的信号。

for (int ii=1;ii<=64;ii++) signal(ii,SIG\_IGN);

这么做的目的是不希望程序被干扰。然后，再设置程序员关心的信号的处理函数。

程序在运行的进程中，如果按Ctrl+c，将向程序发出SIGINT信号，编号是2。

采用“kill 进程编号”或“killall 程序名”向程序发出的是SIGTERM信号，编号是15。

采用“kill -9 进程编号”向程序发出的是SIGKILL信号，编号是9，此信号不能被忽略，也无法捕获，程序将突然死亡。

设置SIGINT和SIGTERM两个信号的处理函数，这两个信号可以使用同一个处理函数，函数的代码是释放资源。

/\*

\* 本程序演示信号的用法。

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <signal.h>

void EXIT(int sig)

{

printf("收到了信号%d，程序退出。\n",sig);

// 在这里添加释放资源的代码

exit(0); // 程序退出。

}

int main()

{

for (int ii=1;ii<=64;ii++) signal(ii,SIG\_IGN); // 屏蔽全部的信号

signal(SIGINT,EXIT); signal(SIGTERM,EXIT); // 设置SIGINT和SIGTERM的处理函数

while (true) // 每隔一秒执行一次任务。

{

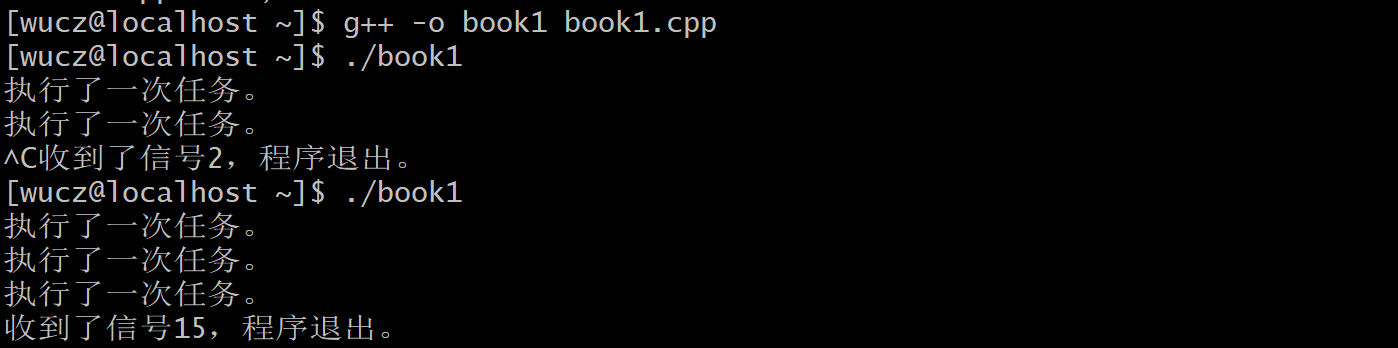
printf("执行了一次任务。\n");

sleep(1);

}

}

**运行效果**



不管是用Ctrl+c还是kill，程序都能体面的退出。

## 6、发送信号

Linux操作系统提供了kill和killall命令向程序发送信号，C语言也提供了kill库函数，用于在程序中向其它进程或者线程发送信号。

函数声明：

int kill(pid\_t pid, int sig);

kill函数将参数sig指定的信号给参数pid 指定的进程。

参数pid 有几种情况：

1）pid>0 将信号传给进程号为pid 的进程。

2）pid=0 将信号传给和目前进程相同进程组的所有进程，常用于父进程给子进程发送信号，注意，发送信号者进程也会收到自己发出的信号。

3）pid=-1 将信号广播传送给系统内所有的进程，例如系统关机时，会向所有的登录窗口广播关机信息。

sig：准备发送的信号代码，假如其值为零则没有任何信号送出，但是系统会执行错误检查，通常会利用sig值为零来检验某个进程是否仍在运行。

返回值说明： 成功执行时，返回0；失败返回-1，errno被设为以下的某个值。

EINVAL：指定的信号码无效（参数 sig 不合法）。

EPERM：权限不够无法传送信号给指定进程。

ESRCH：参数 pid 所指定的进程或进程组不存在。