###### 1.概念

信号(signal)是Linux进程间通信的一种机制，全称为软中断信号，也被称为软中断。信号本质上是在软件层次上对硬件中断机制的一种模拟。

与其他进程间通信方式（例如管道、共享内存等）相比，信号所能传递的信息比较粗糙，只是一个整数。但正是由于传递的信息量少，信号也便于管理和使用，可以用于系统管理相关的任务，例如通知进程终结、中止或者恢复等。

每种信号用一个整型常量宏表示，以SIG开头，比如SIGCHLD、SIGINT等，它们在系统头文件<signal.h>中定义。

信号是异步的，一个进程不必通过任何操作来等待信号的到达，因为，事实上，进程也不知道信号具体什么时候会到达。信号是进程间通信机制中唯一的异步通信机制。

实现原理上，一个进程收到一个信号与处理器收到一个中断请求可以说是一样的。

2.产生方式

信号由内核(kernel)管理，产生方式有：

（1）可以由内核自身产生，比如出现硬件错误、内存读取错误，分母为0的除法等，内核需要通知相应进程。

（2）也可以由其他进程产生并发送给内核，再由内核传递给目标进程。

###### 3.信号传递的过程

"生成（raise）"表示产生一个信号，"捕获（catch）"表示接收到一个信号，信号可以被生成，捕获，响应或（至少对一些信号）忽略。

内核中针对每一个进程都有一个表来保存信号。当内核需要将信号传递给某个进程时，就在该进程对应的表中写入信号，这样就生成了信号。当该进程由用户态陷入内核态，再次切换到用户态之前，会查看表中的信号。如果有信号，进程就会首先执行信号对应的操作，此时叫做执行信号。从生成信号到将信号传递给对应进程这段时间，信号处于等待状态。

我们可以编写代码，让进程阻塞(block)某些信号，也就是让这些信号始终处于等待的状态，直到进程取消阻塞(unblock)或者忽略信号。

###### 4.分类

（1）可靠信号和不可靠信号

"不可靠信号"：早期Unix系统中的信号机制比较简单和原始，后来在实践中暴露出一些问题，

因此，把那些建立在早期机制上的信号叫做"不可靠信号"，信号值小于SIGRTMIN（Redhat7.2中，SIGRTMIN=32，SIGRTMAX=63）的信号都是不可靠信号。它的主要问题是：

A.进程每次处理信号后，就将对信号的响应设置为默认动作。在某些情况下，将导致对信号的错误处理；如果不希望这样的操作，那么就要在信号处理函数结尾再一次调用signal，重新安装该信号。

B.信号可能丢失。

Linux支持不可靠信号，但是对不可靠信号机制做了改进。在调用完信号处理函数后，不必重新调用该信号的安装函数（信号安装函数是在可靠机制上的实现）。

因此，Linux下的不可靠信号问题主要指的是信号可能丢失。

"可靠信号"：由于原来定义的信号已有许多应用，不好再做改动，最终只好又新增加了一些信号，并在一开始就把它们定义为可靠信号，这些信号支持排队，不会丢失。同时，信号的发送和安装也出现了新版本：信号发送函数sigqueue及信号安装函数sigaction。信号值位于SIGRTMIN和SIGRTMAX之间的信号都是可靠信号。

Linux在支持新版本的信号安装函数sigation以及信号发送函数sigqueue的同时，仍然支持早期的signal信号安装函数，和信号发送函数kill。

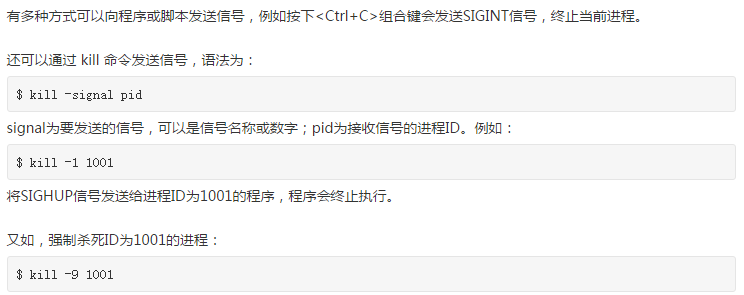
对于signal及sigaction来说，它们都不能把SIGRTMIN以前的信号变成可靠信号（都不支持排队，仍有可能丢失，仍然是不可靠信号），而且对SIGRTMIN以后的信号都支持排队。这两个函数的最大区别在于：经过sigaction安装的信号都能传递信息给信号处理函数（对所有信号这一点都成立），经过signal安装的信号却不能传递信息给信号处理函数。对于信号发送函数来说也是一样的。

（2）实时信号与非实时信号

非实时信号都不支持排队，都是不可靠信号；实时信号都支持排队，都是可靠信号。

5.信号的发送

（1）linux中



（2）c语言编程时

发送信号的主要函数有：kill、raise、sigqueue、alarm、setitimer以及abort。



sig：是信号值，当为0时（即空信号），实际不发送任何信号，但照常进行错误检查.因此，可用于检查目标进程是否存在，以及当前进程是否具有向目标发送信号的权限（root权限的进程可以向任何进程发送信号，非root权限的进程只能向属于同一个session或者同一个用户的进程发送信号）。

kill函数把参数sig给定的信号发送结由参数pid给出的进程号所指定的进程，Kill最常用于"pid>0"时的信号发送，调用成功，返回0；否则，返回-1。

6.进程对信号的响应

进程可以通过三种方式来响应一个信号：

（1）忽略信号

即对信号不做任何处理，其中，有两个信号不能忽略：SIGKILL及SIGSTOP；

（2）捕捉信号

定义信号处理函数，当信号发生时，执行相应的处理函数；

（3）执行缺省操作，

Linux对每种信号都规定了默认操作，详细情况请参考相关资料。注意，进程对实时信号的缺省反应是进程终止。

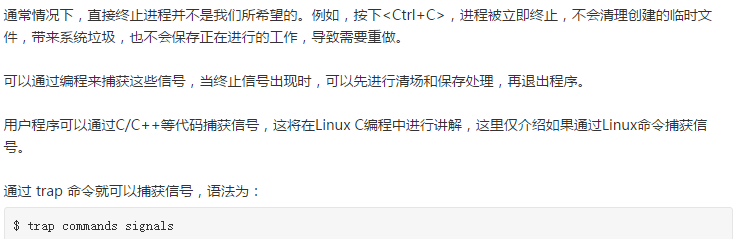
Linux究竟采用上述三种方式的哪一个来响应信号，取决于传递给相应API函数的参数。



函数ouch对通过参数sig传递进来的信号作出响应。信号出现时，程序调用该函数，它打印一条消息，然后将信号SIGINT（默认情况下，按ctrl+c产生这个信号）的处理方式恢复为默认行为。



linux中：







6.信号的安装（设置信号关联动作）

如果进程要处理某一信号，那么就要在进程中安装该信号。安装信号主要用来确定信号值及进程针对该信号值的动作之间的映射关系，即进程将要处理哪个信号；该信号被传递给进程时，将执行何种操作。

linux主要有两个函数实现信号的安装：signal、sigaction。

signal：在可靠信号系统调用的基础上实现，是库函数。它只有两个参数，不支持信号传递信息，主要是用于前32种非实时信号的安装；



signum：指定信号的值，handler：指定针对前面信号值的处理，可以忽略该信号（参数设为SIG\_IGN）；可以采用系统默认方式处理信号（参数设为SIG\_DFL）；也可以自己实现处理方式（参数指定一个函数地址）。

如果signal调用成功，返回最后一次为安装信号signum而调用signal时的handler值；失败则返回SIG\_ERR。

sigaction：是较新的函数（由两个系统调用实现：sys\_signal以及sys\_rt\_sigaction），有三个参数，支持信号传递信息，主要用来与sigqueue系统调用配合使用，当然，sigaction同样支持非实时信号的安装。sigaction优于signal主要体现在支持信号带有参数。



sigaction函数用于改变进程接收到特定信号后的行为。

sig：为信号的值，可以为除SIGKILL及SIGSTOP外的任何一个特定有效的信号为这两个信号定义自己的处理函数，将导致信号安装错误。

act：指向结构体sigaction的一个实例的指针，在结构体sigaction的实例中，指定了对特定信号的处理，可以为空，进程会以缺省方式对信号处理；参数act最为重要，其中包含了对指定信号的处理、信号所传递的信息、信号处理函数执行过程中应屏蔽掉哪些函数等等.

oldact：指向的对象用来保存原来对相应信号的处理，可指定oldact为NULL。如果oldact不是空指针，sigaction将把原先对该信号的动作写到它指向的位置。如果oldact是空指针，则sigaction函数就不需要再做其它设置了。如果把参数act，oldact都设为NULL，那么该函数可用于检查信号的有效性。

7.信号阻塞与信号未决

每个进程都有一个用来描述哪些信号递送到进程时将被阻塞的信号集，该信号集中的所有信号在递送到进程后都将被阻塞。

8.信号生命周期

从信号发送到信号处理函数的执行完毕。对于一个完整的信号生命周期（从信号发送到相应的处理函数执行完毕）来说，可以分为三个重要的阶段，信号诞生，信号在进程中注册完毕，信号在进程中的注销完毕，信号处理函数执行完毕。相邻两个事件的时间间隔构成信号生命周期的一个阶段。

linux下的信号应用，程序员所要做的最多只有三件事情：安装信号（推荐使用sigaction），实现三参数信号处理函数（handler），发送信号（推荐使用sigqueue）。

9.信号应用实例

linux常用的信号类型有：

1）SIGHUP

在用户终端连接（正常或非正常）结束时发出，通常是在终端的控制进程结束时，通知同一session内的各个作业，这时它们与控制终端不再关联。

登录Linux时，系统会分配给登录用户一个终端（Session）。在这个终端运行的所有程序，包括前台进程组和后台进程组，一般都属于这个Session。当用户退出Linux登录时，前台进程组和后台有对终端输出的进程将会收到SIGHUP信号。这个信号的默认操作为终止进程，因此前台进程组和后台有终端输出的进程就会中止。不过可以捕获这个信号，比如wget能捕获SIGHUP信号，并忽略它，这样就算退出了Linux登录，wget也能继续下载。

此外，对于与终端脱离关系的守护进程，这个信号用于通知它重新读取配置文件。

2）SIGINT

程序终止（interrupt）信号，在用户键入INTR字符（通常是Ctrl-C）时发出，用于通知前台进程组终止进程。

9）SIGKILL

用来立即结束程序的运行。本信号不能被阻塞、处理和忽略。如果管理员发现某个进程终止不了，可尝试发送这个信号。