[linux系统编程之信号（一）：中断与信号](http://www.cnblogs.com/mickole/p/3189156.html)

**一，什么是中断？**

**1．中断的基本概念**

中断是指计算机在执行期间，系统内发生任何非寻常的或非预期的急需处理事件，使得CPU暂时中断当前正在执行的程序而转去执行相应的事件处理程序，待处理完毕后又返回原来被中断处继续执行或调度新的进程执行的过程。引起中断发生的事件被称为中断源。中断源向CPU发出的请求中断处理信号称为中断请求，而CPU收到中断请求后转到相应的事件处理程序称为中断响应。

在有些情况下，尽管产生了中断源和发出了中断请求，但CPU内部的处理器状态字PSW的中断允许位已被清除，从而不允许CPU响应中断。这种情况称为禁止中断。CPU禁止中断后只有等到PSW的中断允许位被重新设置后才能接收中断。禁止中断也称为关中断，PSW的中断允许位的设置也被称为开中断。开中断和关中断是为了保证某段程序执行的原子性。

还有一个比较常用的概念是中断屏蔽。中断屏蔽是指在中断请求产生之后，系统有选择地封锁一部分中断而允许另一部分中断仍能得到响应。不过，有些中断请求是不能屏蔽甚至不能禁止的，也就是说，这些中断具有最高优先级，只要这些中断请求一旦提出，CPU必须立即响应。例如，电源掉电事件所引起的中断就是不可禁止和不可屏蔽的。

**2．中断的分类与优先级**

根据系统对中断处理的需要，操作系统一般对中断进行分类并对不同的中断赋予不同的处理优先级，以便在不同的中断同时发生时，按轻重缓急进行处理。

根据中断源产生的条件，可把中断分为外中断和内中断。外中断是指来自处理器和内存外部的中断，包括I／0设备发出的I／O中断、外部信号中断(例如用户键人ESC键)。各种定时器引起的时钟中断以及调试程序中设置的断点等引起的调试中断等。外中断在狭义上一般被称为中断。

内中断主要指在处理器和内存内部产生的中断。内中断一般称为陷阱(trap)或异常。它包括程序运算引起的各种错误，如地址非法、校验错、页面失效、存取访问控制错、算术操作溢出、数据格式非法、除数为零、非法指令、用户程序执行特权指令、分时系统中的时间片中断以及从用户态到核心态的切换等都是陷阱的例子。

为了按中断源的轻重缓急处理响应中断，操作系统为不同的中断赋予不同的优先级。例如在UNIX系统中，外中断和陷阱的优先级共分为8级。为了禁止中断或屏蔽中断，CPU的处理器状态字PSW中也设有相应的优先级。如果中断源的优先级高于PSW的优先级，则CPU响应该中断源的请求；反之，CPU屏蔽该中断源的中断请求。

各中断源的优先级在系统设计时给定，在系统运行时是固定的。而处理器的优先级则根据执行情况由系统程序动态设定。

除了在优先级的设置方面有区别之外，中断和陷阱还有如下主要区别：

陷阱通常由处理器正在执行的现行指令引起，而中断则是由与现行指令无关的中断源引起的。陷阱处理程序提供的服务为当前进程所用，而中断处理程序提供的服务则不是为了当前进程的。

CPU执行完一条指令之后，下一条指令开始之前响应中断，而在一条指令执行中也可以响应陷阱。例如执行指令非法时，尽管被执行的非法指令不能执行结束，但CPU仍可对其进行处理。

**3．软中断**

软中断的概念主要来源于UNIX系统。软中断是对应于硬中断而言的。通过硬件产生相应的中断请求，称为硬中断。而软中断则不然，它是在通信进程之间通过模拟硬中断而实现的一种通信方式。中断源发出软中断信号后，CPU或者接收进程在“适当的时机”进行中断处理或者完成软中断信号所对应的功能。这里“适当的时机”，表示接收软中断信号的进程须等到该接收进程得到处理器之后才能进行。如果该接收进程是占据处理器的，那么，该接收进程在接收到软中断信号后将立即转去执行该软中断信号所对应的功能。

**4．中断处理过程**

一旦CPU响应中断，转人中断处理程序，系统就开始进行中断处理。下面对中断处理过程进行详细说明：

1)CPU检查响应中断的条件是否满足。CPU响应中断的条件是：有来自于中断源的中断请求、CPU允许中断。如果中断响应条件不满足，则中断处理无法进行。

2)如果CPU响应中断，则CPU关中断，使其进入不可再次响应中断的状态。

3)保存被中断进程现场。为了在中断处理结束后能使进程正确地返回到中断点，系统必须保存当前处理器状态字PSW和程序计数器PC等的值。这些值一般保存在特定堆栈或硬件寄存器中。

4)分析中断原因，调用中断处理子程序。在多个中断请求同时发生时，处理优先级最高的中断源发出的中断请求。在系统中，为了处理上的方便，通常都是针对不同的中断源编制有不同的中断处理子程序(陷阱处理子程序)。这些子程序的人口地址(或陷阱指令的人口地址)存放在内存的特定单元中。再者，不同的中断源也对应着不同的处理器状态字PSW。这些不同的PSW被放在相应的内存单元中，与中断处理子程序人口地址一起构成中断向量。显然，根据中断或陷阱的种类，系统可由中断向量表迅速地找到该中断响应的优先级、中断处理子程序(或陷阱指令)的入口地址和对应的PSW。

5)执行中断处理子程序。对陷阱来说，在有些系统中则是通过陷阱指令向当前执行进程发出软中断信号后调用对应的处理子程序执行。

6)退出中断，恢复被中断进程的现场或调度新进程占据处理器。

7)开中断，CPU继续执行。

**5．设备管理程序与中断方式**

处理器的高速和输入输出设备低速之间的矛盾，是设备管理要解决的一个重要问题。为了提高整体效率，减少在程序直接控制方式中的CPU等待时间以及提高系统的并行工作效率，采用中断方式来控制输入输出设备和内存与CPU之间的数据传送，是很有必要的。

在硬件结构上，这种方式要求CPU与输入输出设备(或控制器)之间有相应的中断请求线，而且在输入输出设备控制器的控制状态寄存器上有相应的中断允许位。

**二，什么是信号？**

**1、信号及信号来源**

**信号本质**

信号是在软件层次上对中断机制的一种模拟，在原理上，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。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，事实上，进程也不知道信号到底什么时候到达。

信号是进程间通信机制中唯一的异步通信机制，可以看作是异步通知，通知接收信号的进程有哪些事情发生了。信号机制经过POSIX实时扩展后，功能更加强大，除了基本通知功能外，还可以传递附加信息。

**信号来源**

信号事件的发生有两个来源：硬件来源(比如我们按下了键盘或者其它硬件故障)；软件来源，最常用发送信号的系统函数是kill, raise, alarm和setitimer以及sigqueue函数，软件来源还包括一些非法运算等操作。

**2、信号的种类**

可以从两个不同的分类角度对信号进行分类：（1）可靠性方面：可靠信号与不可靠信号；（2）与时间的关系上：实时信号与非实时信号。在《Linux环境进程间通信（一）：管道及有名管道》的附1中列出了系统所支持的所有信号。

（一）、可靠信号与不可靠信号

**"不可靠信号"**

Linux信号机制基本上是从Unix系统中继承过来的。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(Red hat 7.2中，SIGRTMIN=32，SIGRTMAX=63)的信号都是不可靠信号。这就是"不可靠信号"的来源。它的主要问题是：

* 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；因此，用户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。
* 信号可能丢失，后面将对此详细阐述。   
  因此，早期unix下的不可靠信号主要指的是进程可能对信号做出错误的反应以及信号可能丢失。

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"**

随着时间的发展，实践证明了有必要对信号的原始机制加以改进和扩充。所以，后来出现的各种Unix版本分别在这方面进行了研究，力图实现"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()。POSIX.4对可靠信号机制做了标准化。但是，POSIX只对可靠信号机制应具有的功能以及信号机制的对外接口做了标准化，对信号机制的实现没有作具体的规定。

信号值位于SIGRTMIN和SIGRTMAX之间的信号都是可靠信号，可靠信号克服了信号可能丢失的问题。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，支持信号发送函数kill()。

注：不要有这样的误解：由sigqueue()发送、sigaction安装的信号就是可靠的。事实上，可靠信号是指后来添加的新信号（信号值位于SIGRTMIN及SIGRTMAX之间）；不可靠信号是信号值小于SIGRTMIN的信号。信号的可靠与不可靠只与信号值有关，与信号的发送及安装函数无关。目前linux中的signal()是通过sigation()函数实现的，因此，即使通过signal（）安装的信号，在信号处理函数的结尾也不必再调用一次信号安装函数。同时，由signal()安装的实时信号支持排队，同样不会丢失。

对于目前linux的两个信号安装函数: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），而经过signal安装的信号却不能向信号处理函数传递信息。对于信号发送函数来说也是一样的。

（二）、实时信号与非实时信号

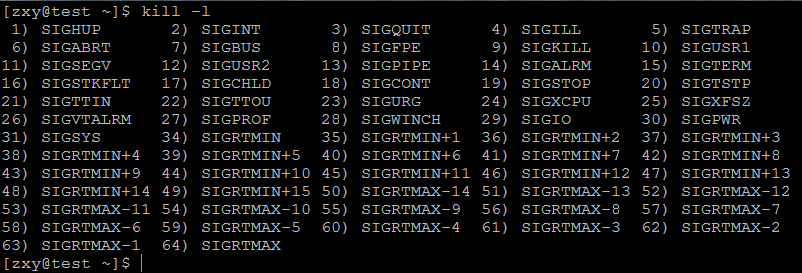
早期Unix系统只定义了32种信号，Ret hat7.2支持64种信号，编号0-63(SIGRTMIN=31，SIGRTMAX=63)，将来可能进一步增加，这需要得到内核的支持。前32种信号已经有了预定义值，每个信号有了确定的用途及含义，并且每种信号都有各自的缺省动作。如按键盘的CTRL ^C时，会产生SIGINT信号，对该信号的默认反应就是进程终止。后32个信号表示实时信号，等同于前面阐述的可靠信号。这保证了发送的多个实时信号都被接收。实时信号是POSIX标准的一部分，可用于应用进程。

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。

（三）,信号名称

查看linux所支持的信号可用：kill –l

共64种：

[](http://images.cnitblog.com/blog/529981/201307/14104400-0541f2d712e04ca793a4dbeecf902d1e.png)

[](http://images.cnitblog.com/blog/529981/201307/14104401-e14c06f2ab494f408848ce388450a2df.png)

[](http://images.cnitblog.com/blog/529981/201307/14104402-436280b09f8d48f689cbbf003f8ade82.png)

**3****、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[2]以及其它资料。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。

注：本文参考：<http://blog.csdn.net/lmh12506/article/details/6681663>和

<http://blog.csdn.net/johnny710vip/article/details/6990514>