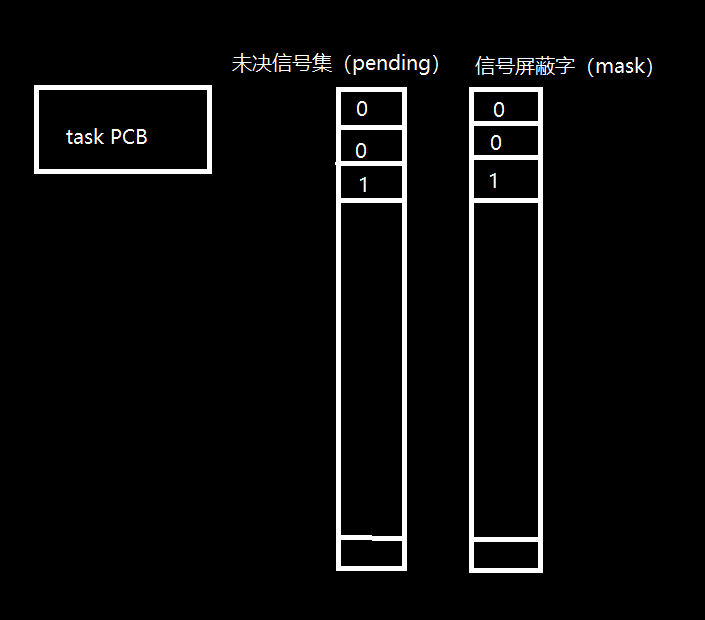
信号集、信号的阻塞

## 有时候不希望在接到信号时就立即停止当前执行，去处理信号，同时也不希望忽略该信号，而是延时一段时间去调用信号处理函数。这种情况可以通过阻塞信号实现。

**信号的阻塞概念**：信号的”阻塞“是一个开关动作，指的是阻止信号被处理，但不是阻止信号产生。



信号的状态：

信号递达（Delivery ）：实际信号执行的处理过程(3种状态：忽略，执行默认动作，捕获)

信号未决（Pending）：从产生到递达之间的状态

## 信号集操作函数

sigset\_t set; 自定义信号集。 是一个32bit 64bit 128bit的数组。

sigemptyset(sigset\_t \*set); 清空信号集

sigfillset(sigset\_t \*set); 全部置1

sigaddset(sigset\_t \*set, int signum); 将一个信号添加到集合中

sigdelset(sigset\_t \*set, int signum); 将一个信号从集合中移除

sigismember（const sigset\_t \*set，int signum); 判断一个信号是否在集合中。

设定对信号集内的信号的处理方式(阻塞或不阻塞)

#include <signal.h>

int sigprocmask( int how, const sigset\_t \*restrict set, sigset\_t \*restrict oset );

返回值：若成功则返回0，若出错则返回-1

首先，若oset是非空指针，那么进程的当前信号屏蔽字通过oset返回。

其次，若set是一个非空指针，则参数how指示如何修改当前信号屏蔽字。

how可选用的值：（注意，不能阻塞SIGKILL和SIGSTOP信号）

SIG\_BLOCK ： 把参数set中的信号添加到信号屏蔽字中

SIG\_UNBLOCK： 从信号屏蔽字中删除参数set中的信号

SIG\_SETMASK： 把信号屏蔽字设置为参数set中的信号

int pause(void);

进程一直阻塞，直到被信号中断，返回值：-1 并设置errno为EINTR

函数行为：

1如果信号的默认处理动作是终止进程，则进程终止，pause函数么有机会返回。

2如果信号的默认处理动作是忽略，进程继续处于挂起状态，pause函数不返回

3 如果信号的处理动作是捕捉，则调用完信号处理函数之后，pause返回-1。

4 pause收到的信号如果被屏蔽，那么pause就不能被唤醒

int sigsuspend(const sigset\_t \*sigmask);

功能：将进程的屏蔽字替换为由参数sigmask给出的信号集，然后挂起进程的执行

参数：

sigmask：希望屏蔽的信号