信号的产生

1.通过终端按键产生信号

2.通过系统函数向进程发信号

- kill

3.由软件产生信号

- alarm

POSIX 信号和多线程应用

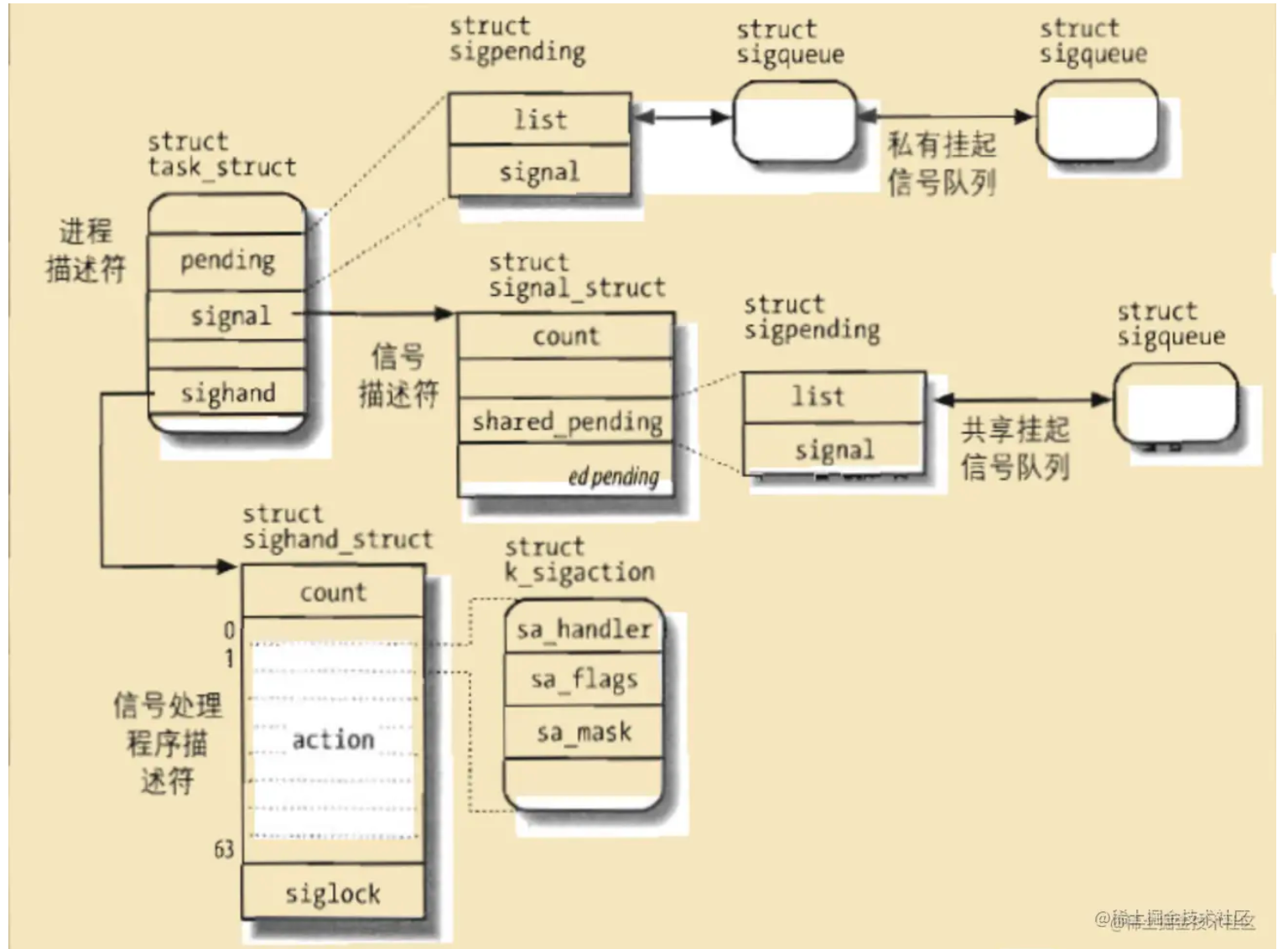
1.信号处理程序必须在多线程应用的所有线程之间共享,不过,每个线程都必须有自己的挂起信号掩码和阻塞信号掩码。

2.POSIX 库函数 kill() / sigqueue() 必须向所有的多线程应用而不是某个特殊的线程发送信号。所有由内核产生的信号也是如此(SIGQUIT等)

3.每个发送给多线程应用的信号仅传送给一个线程,这个线程是由内核在从不会阻塞该信号的线程中随意选择出来的。

4.如果向多线程应用发送一个致命的信号,那么内核将杀死该应用的所有线程,而不仅仅是杀死接收信号的那个线程。

与信号相关的数据结构



1.signal 信号描述符的指针

2.sighand 信号处理程序描述符的指针「每一个进程引用一个信号处理程序描述符」

3.信号处理程序描述符就可以由几个进程共享。

4.线程组中的所有轻量级进程都应用相同的信号描述符和信号处理程序描述符。

sigaction

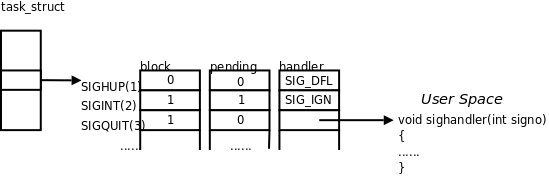
int sigaction(int sig, const struct sigaction \*restrict act, struct sigaction \*restrict oact);

该函数为某一信号设置全局信号处理程序,其它线程的信号处理程序就会被覆盖。「使用完需要恢复之前的信号处理」

sigwait

在调用 sigwait() 时，set 指定的信号应该被阻止，但不能被忽略。对忽略的信号调用 sigwait() 的进程将无限期地等待。

阻塞信号



捕捉信号

