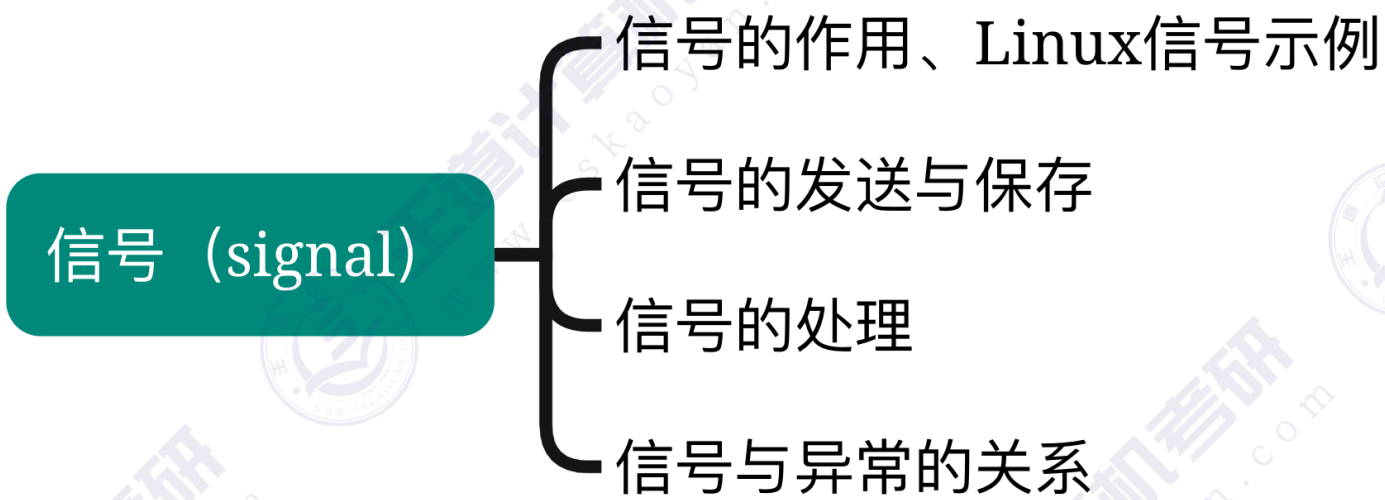
本节内容

信号

知识总览



信号量（Semaphore）—— 实现进程间的同步、互斥

信号（Signal）—— 实现进程间通信（IPC，Inter Process Communication）

信号的作用

信号（signal）：用于通知进程某个特定事件已经发生。进程收到一个信号后，对该信号进行处理。

Linux 操作系统定义的 30 种信号类型

序号	信号名	信号对应的事件	默认处理
1	SIGHUP	…	…
2	SIGINT	来自键盘的中断（Control-C）	终止
3	SIGQUIT	…	…
…	…	…	…
8	SIGFPE	浮点异常（如：除以0）	终止并转储内存
9	SIGKILL	杀死进程（如：父进程杀子进程）	终止
…	…	…	…
28	SIGWINCH	窗口大小变化	忽略
…	…	…	…
30	SIGPWR	…	…

注1：不同的操作系统对信号类型的定义不一样

注2：通常用宏定义常量表示信号名，如：#define SIGINT 2

Ref：《深入理解计算机系统》

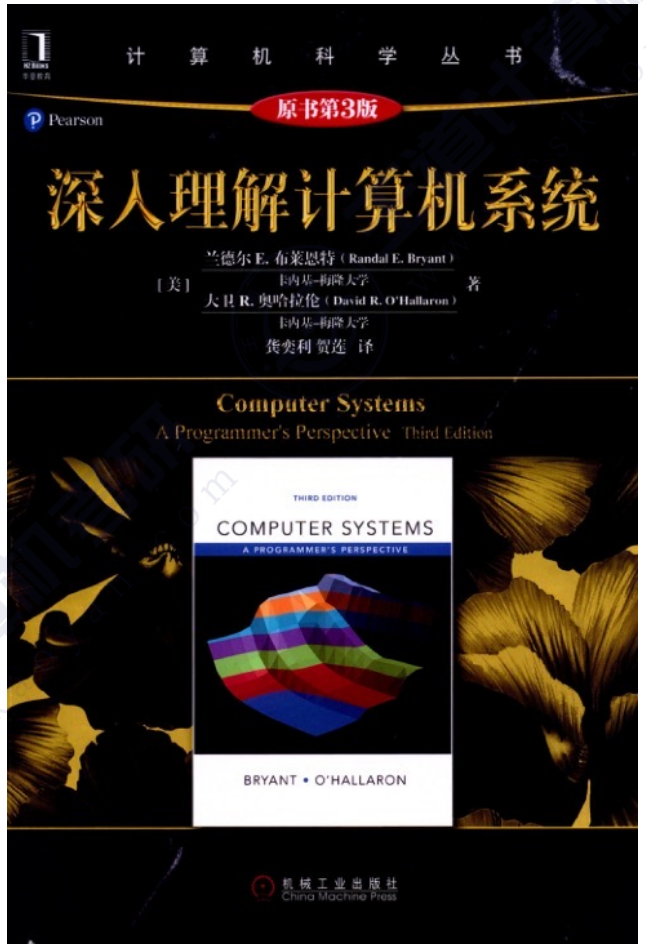
序号	名称	默认行为	相应事件
1	SIGHUP	终止	终端线挂断
2	SIGINT	终止	来自键盘的中断
3	SIGQUIT	终止	来自键盘的退出
4	SIGILL	终止	非法指令
5	SIGTRAP	终止并转储内存①	跟踪陷阱
6	SIGABRT	终止并转储内存①	来自 abort 函数的终止信号
7	SIGBUS	终止	总线错误
8	SIGFPE	终止并转储内存①	浮点异常
9	SIGKILL	终止②	杀死程序
10	SIGUSR1	终止	用户定义的信号 1
11	SIGSEGV	终止并转储内存①	无效的内存引用（段故障）
12	SIGUSR2	终止	用户定义的信号 2
13	SIGPIPE	终止	向一个没有读用户的管道做写操作
14	SIGALRM	终止	来自 alarm 函数的定时器信号
15	SIGTERM	终止	软件终止信号
16	SIGSTKFLT	终止	协处理器上的栈故障
17	SIGCHLD	忽略	一个子进程停止或者终止
18	SIGCONT	忽略	继续进程如果该进程停止
19	SIGSTOP	停止直到下一个 SIGCONT②	不是来自终端的停止信号
20	SIGTSTP	停止直到下一个 SIGCONT	来自终端的停止信号
21	SIGTTIN	停止直到下一个 SIGCONT	后台进程从终端读
22	SIGTTOU	停止直到下一个 SIGCONT	后台进程向终端写
23	SIGURG	忽略	套接字上的紧急情况
24	SIGXCPU	终止	CPU 时间限制超出
25	SIGXFSZ	终止	文件大小限制超出
26	SIGVTALRM	终止	虚拟定时器期满
27	SIGPROF	终止	剖析定时器期满
28	SIGWINCH	忽略	窗口大小变化
29	SIGIO	终止	在某个描述符上可执行 I/O 操作
30	SIGPWR	终止	电源故障

图 8-26　Linux 信号

注：①多年前，主存是用一种称为磁芯存储器(core memory)的技术来实现的。“转储内存”(dumping core)是一个历史术语，意思是把代码和数据内存段的映像写到磁盘上。

②这个信号既不能被捕获，也不能被忽略。

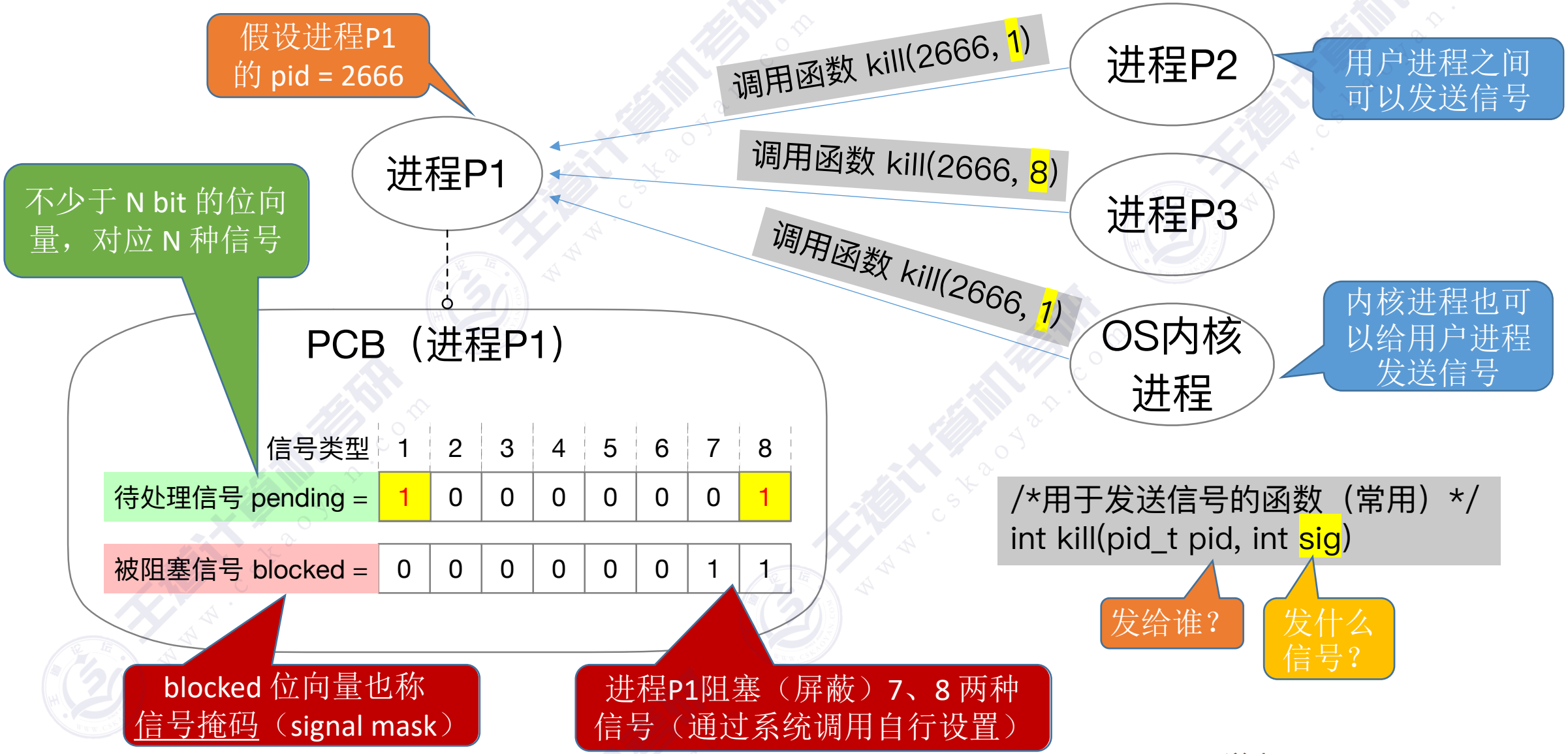
（来源：man 7 signal。数据来自 Linux Foundation。）



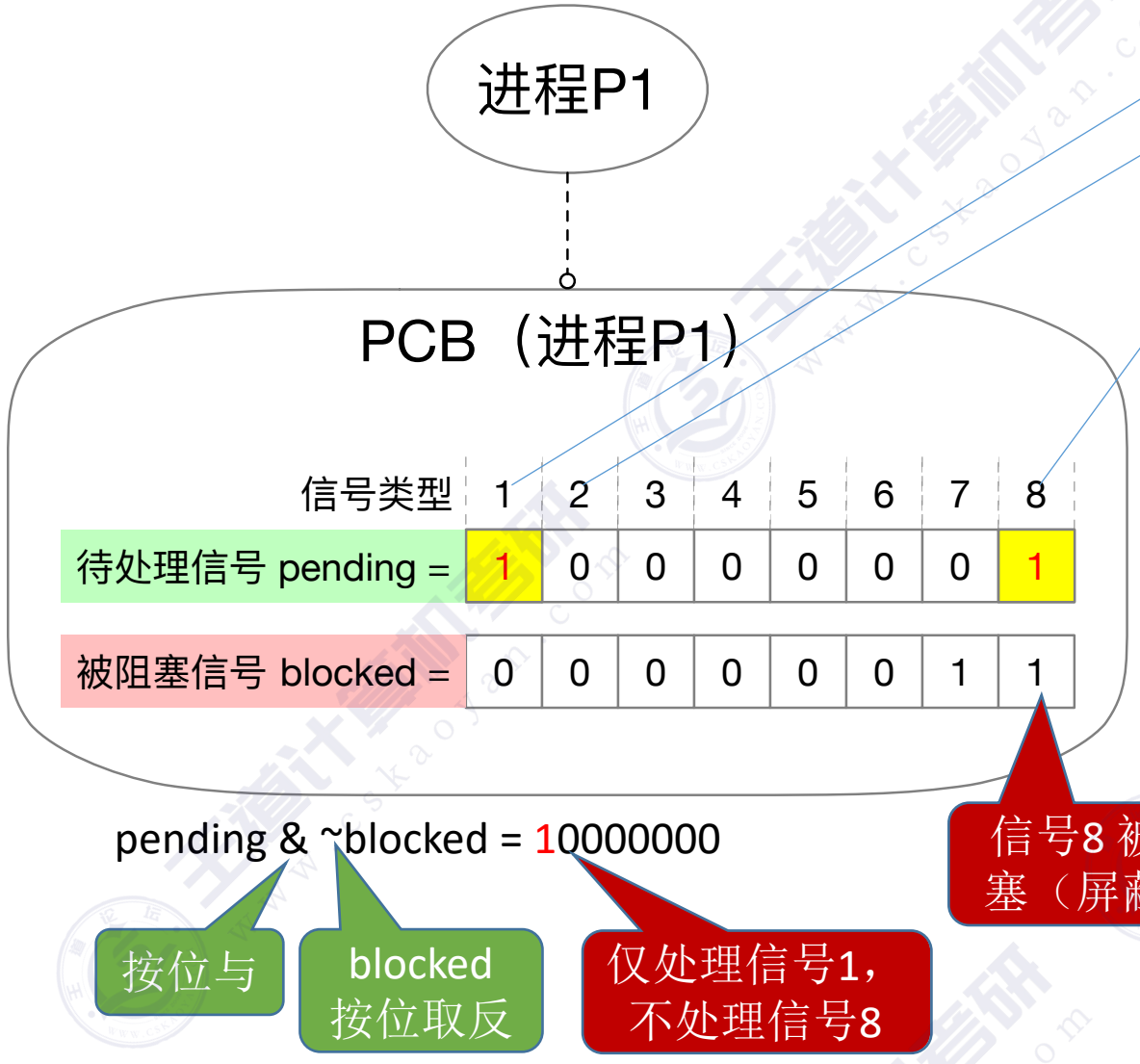
https://man7.org/linux/man-pages/man7/signal.7.html

实现原理——信号的发送与保存

注：进程之间允许发送的信号类型是有限制的



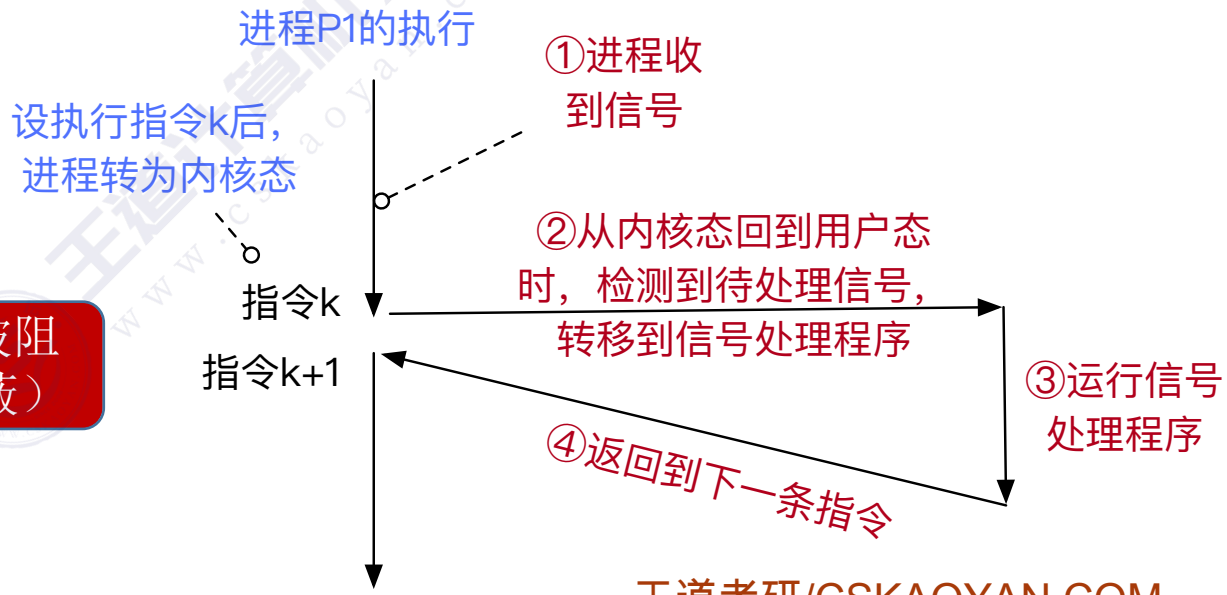
实现原理——信号的处理（When）



信号1的处理程序
信号2的处理程序
.....
信号8的处理程序

When 什么时候处理信号？

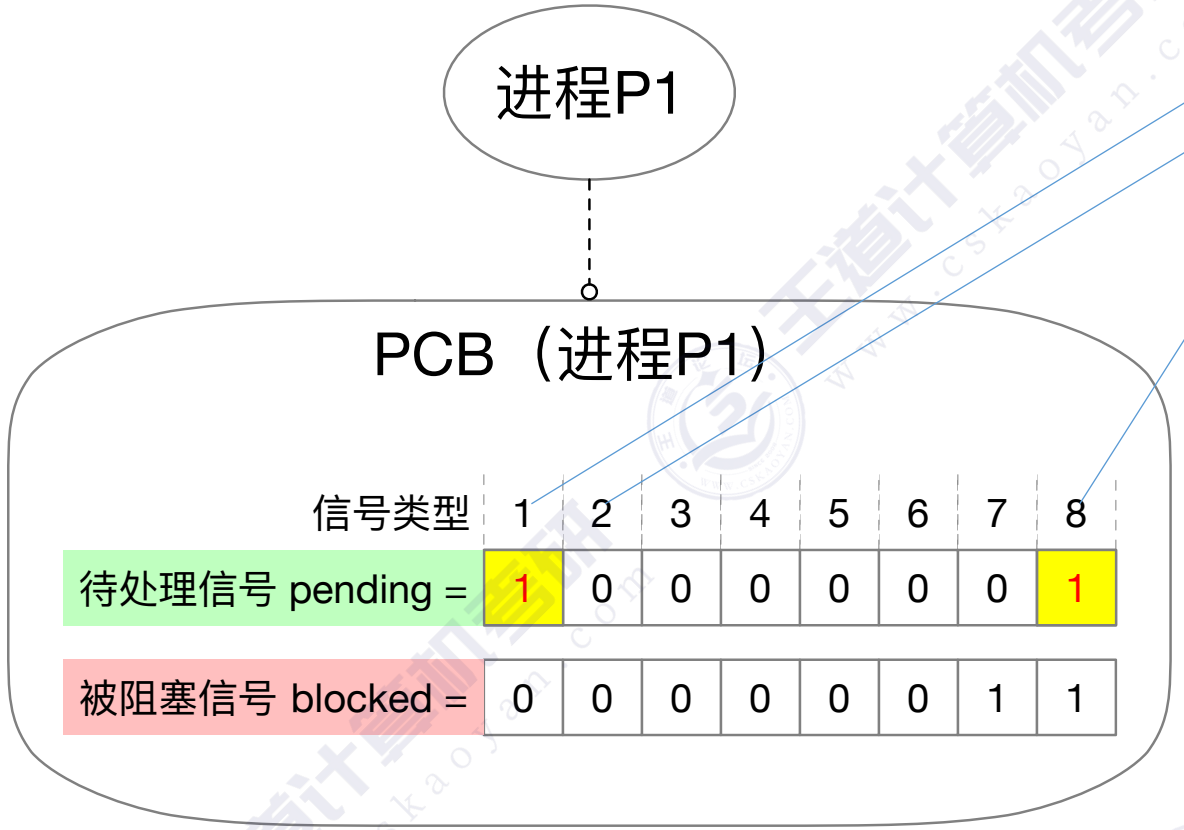
- 当进程从内核态转为用户态时（如：系统调用返回、或中断处理返回时），例行检查是否有待处理信号，如果有，就处理信号。



实现原理——信号的处理（How）

每一种信号都有默认处理程序

sig1_default( ){...} //信号1的默认处理程序
sig2_p1handler( ){...} //进程P1自定义的信号2处理程序
.....
sig8_default( ){...} //信号8的默认处理程序



pending & ~blocked = 10000000

按位与

blocked 按位取反

仅处理信号1，不处理信号8

How 怎么处理信号？

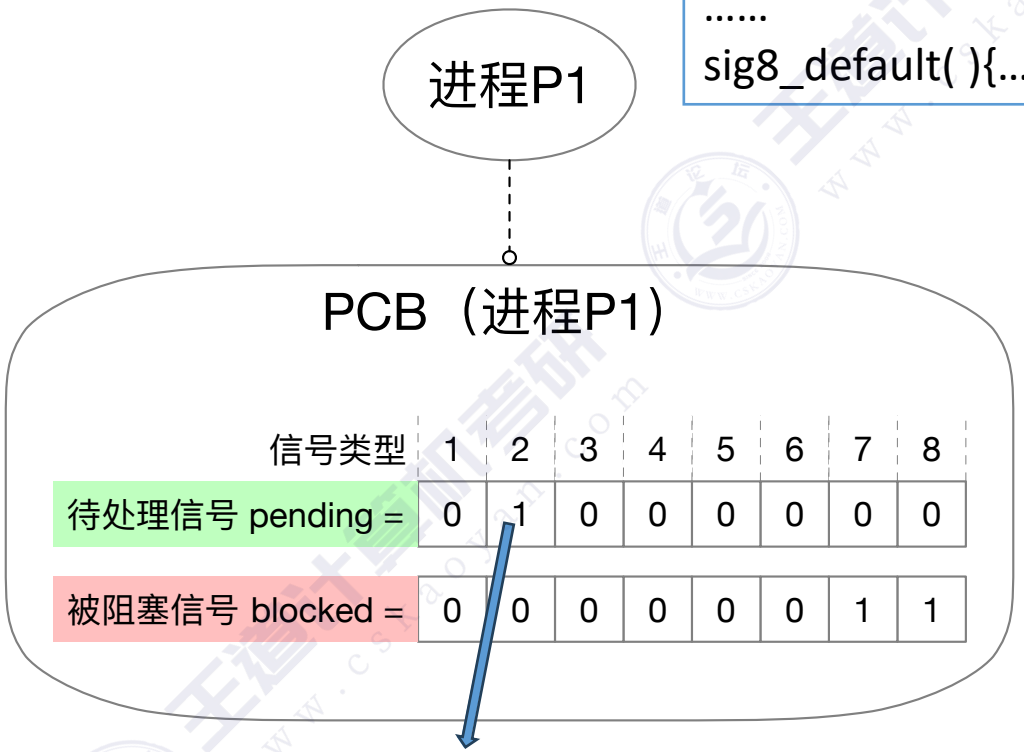
①执行操作系统为此类信号设置的缺省（默认）信号处理程序（某些信号默认忽略，不作处理）

②执行进程为此类信号设置用户自定义信号处理程序（自定义信号处理程序将覆盖①）

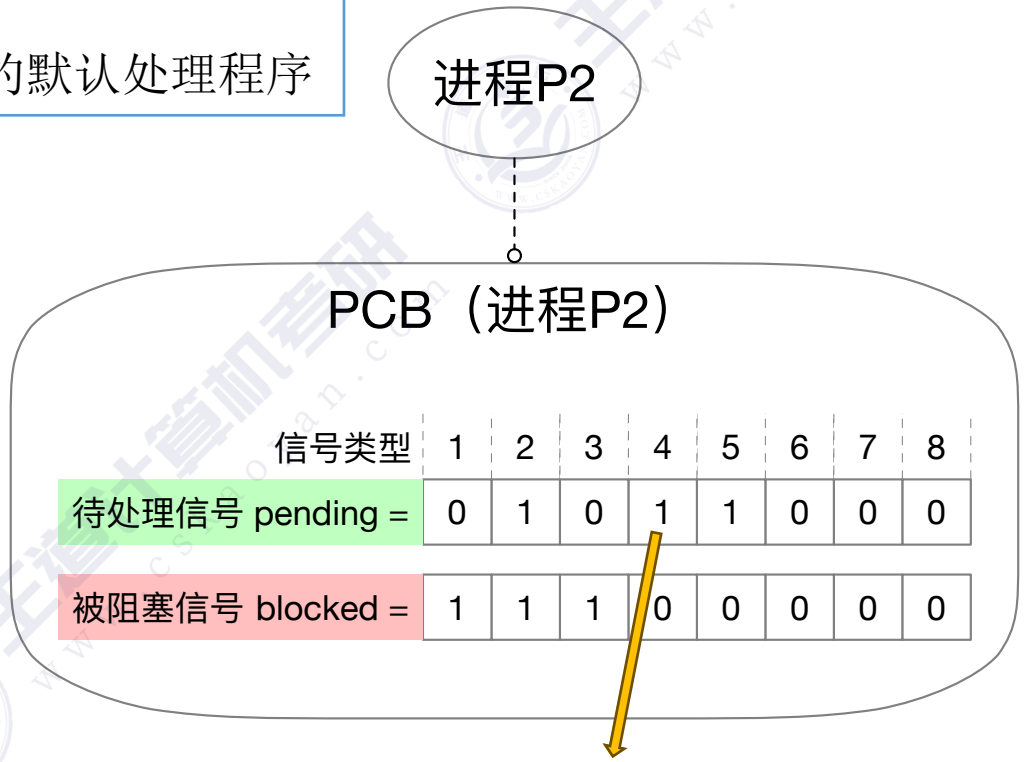
- 信号处理程序运行结束后，通常会返回进程的下一条指令继续执行（除非信号处理程序将进程阻塞或终止）
- 一旦处理了某个信号，就将pending位重置为0
- 重复收到的同类信号，将被简单地丢弃（因为仅有 1bit 记录一类待处理信号）
- 当同时收到多个不同类信号时，通常先处理序号更小的信号。

疑难点1：各个进程的信号处理有何区别？

操作系统规定每种信号的**默认处理程序**：
sig1_default( ){...}　//信号1的默认处理程序
sig2_default( ){...}　//信号2的默认处理程序
……
sig8_default( ){...}　//信号8的默认处理程序



sig2_p1handler( ){...} //进程P1自定义的信号2处理程序



sig4_p2handler( ){...} //进程P2自定义的信号4处理程序

疑难点2：信号与异常有什么关系？

“信号”可以作为“异常”的配套机制，让进程对操作系统的异常处理进行补充。

在进程运行过程中，某些特殊事件可能引发“异常”，操作系统内核负责捕获并处理异常

- 有些异常可以由内核完成全部处理（如：缺页异常），此时就不必再使用信号机制。
- 有些异常无法由内核完成全部处理，可能还需要用户进程配合，此时就可以用“信号机制”与“异常机制”相互配合（如：在Linux中，发生除以0异常时，内核的异常处理程序会向用户进程发送SIGFPE信号。SIGFPE信号的默认处理程序会将进程终止并转储内存；当然进程可以自定义SIGFPE信号处理程序）


不妙

假如：你开发了一个计算器应用，每当用户输入除以0，你的应用就闪退崩溃。这种体验如何？

解决方法：你可以自定义SIGFPE信号处理程序，按照实际需求去决定如何处理


妙呀

知识回顾与重要考点