异常控制流

异常控制流存在于系统的每个层级，最底层的机制称为**异常(Exception)**，用以改变控制流以响应系统事件，通常是由硬件的操作系统共同实现的。更高层次的异常控制流包括**进程切换(Process Context Switch)**、**信号(Signal)**和**非本地跳转(Nonlocal Jumps)**，

Exceptions
把异常交给kernal来应对
异常有它的Exceptions table，每个event被赋予了一个特定的k用于表中寻址

两种异常
Asynchronous Exceptions
example
Timer interrupt
Every few ms, an external timer chip triggers an interrupt
Used by the kernel to take back control from user programs
I/O interrupt from external device
Hitting Ctrl-C at the keyboard
Arrival of a packet from a network
Arrival of data from
interrupt一般返回的是下一条指令

Synchronous Exception
同步异常(Synchronous Exception)是因为执行某条指令所导致的事件，分为陷阱(Trap)、故障(Fault)和终止(Abort)三种情况。
三种类型的处理情况不一样，详细见下图
fault example Page Fault
user想要去写没有初始化后的空间，先copy page from memory，然后返回当前进程，再写。

Process
进程切换
，内存中保存着进程所需的各种信息，因为该进程独占 CPU，所以并不需要保存寄存器值。而在右边的单核多进程模型中，虚线部分可以认为是当前正在执行的进程，因为我们可能会切换到其他进程，所以内存中需要另一块区域来保存当前的寄存器值，以便下次执行的时候进行恢复（也就是所谓的上下文切

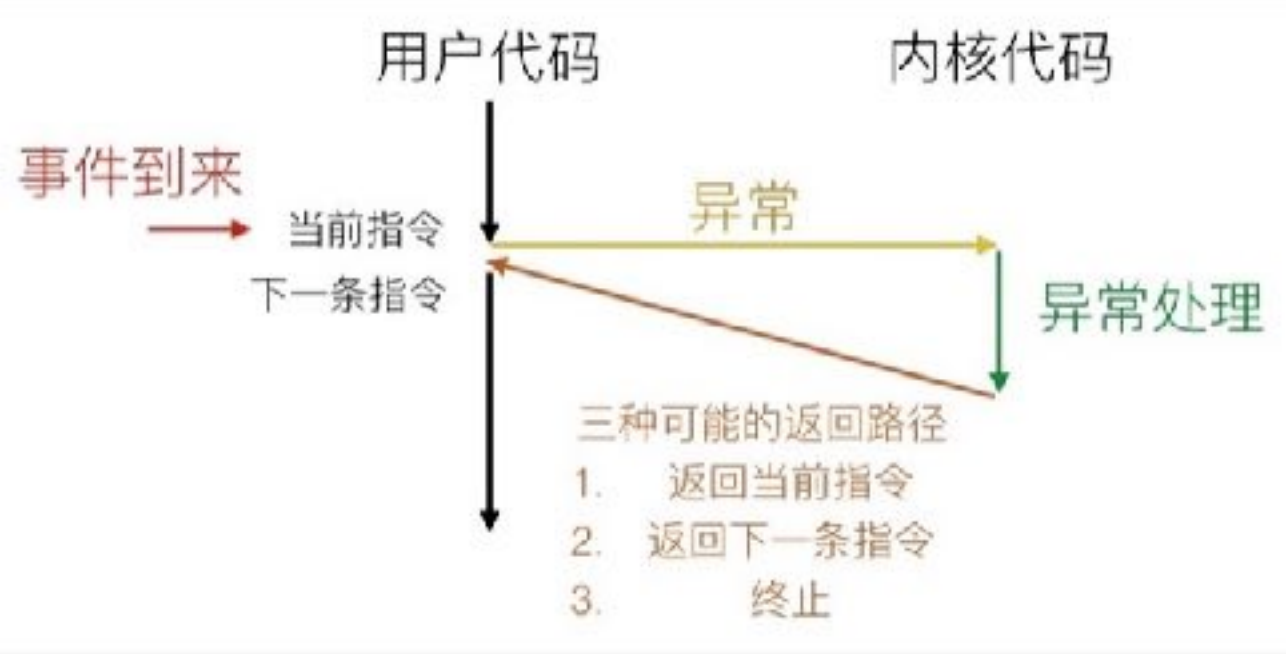
换）。整个过程中，CPU 交替执行不同的进程，虚拟内存系统会负责管理地址空间，而没有执行的进程的寄存器值会被保存在内存中。切换到另一个进程的时候，会载入已保存的对应于将要执行的进程的寄存器值。

FORK函数

进程回收

正常情况下，终止的进程有其父亲进行回收，如果父亲进程死了，而子进程还是zombie情况下，则有init回收

waitpid函数的用法



类型	原因	行为	示例
陷阱	有意的异常	返回到下一条指令	系统调用，断点
故障	潜在可恢复的错误	返回到当前指令	页故障(page faults)
终止	不可恢复的错误	终止当前程序	非法指令