IPC概述

> Inter-Process Communication 进程间通信

为什么要进程间通信?

进程之间不能随意的访问另一个进程的地址空间，但进程之间的确需要进行协作的时候，需要进行数据交互的时候，这个时候在确保进程的独立的同时，还需要进程与进程之间的通信。

通信模型

直接通信

> 进程与进程之间直接进行通信

- 进程必须正确的命令对方
  - `send(P, message)` - 发送消息到进程P （需要知道对方的pid）
  - `receive(Q, message)` - 从进程Q中接收消息（需要知道对方的pid）
- 通信链路的属性
  - 自动建立链路
  - 一条链路恰好对应一对通信进程
  - 每对进程之间只有一个链接存在
  - 链接可以是单向的，但通常是双向的

  没有操作系统的支持很难建立链路

间接通信

> 将消息发送到操作系统指定好的共享区域

- 定向从消息队列接收消息
  - 每个消息队列都有一个唯一ID
  - 只有它们共享了一个消息队列，进程才能够通信
- 通信链路的属性
  - 只有进程共享一个共同的消息队列，才能建立链路
  - 链接可以与许多进程相关联
  - 每队进程可以共享多个通信链路
  - 连接可以是单向或者双向

阻塞通信

如果发送消息没完成，就一直阻塞在那，如果完成了在去干其它的事情

非阻塞通信

发送消息后，不会阻塞在那，不管发送成功与否，都会迅速返回

缓冲

队列的消息可以以这3种方式：

- 0 容量

  发送发必须等待接收方，就像是阻塞通信

- 有量容量 n的优先长度

  队列未满，发送方可以一直发，若队列满了则阻塞，直到队列中的消息被接收后，队列又空出来了才能发

- 无限容量

  发送方不需要等待，可以一直发送

信号

> Signal 信号

- 软件中断通知事件处理
- Example: SIGFPE,SIGKILL.SIGUSER1,SIGSTOP,SIGCONT

接收到信号会发生什么?

1. Catch：编写对应信号值的处理函数，当接收到信号后调用相应的处理函数
2. Ignore: 依靠操作系统的默认操作（即程序不做处理）

   Example: Abort , memory dump , suspend or resume process
3. Mask: 闭塞信号因此不会传送

   可能是暂时的（当处理同样类型的信号）

不足

不能传输要交换的数据，只是起了通知的效果

管道

> 主要进行数据交换

父子进程间的进程交换

列子：ps -ef | grep "a"

竖线 | 就是管道 将ps的输出 定向到了grep的输入 最终达到了从ps的输出当中在grep出a的过滤效果

此时 操作系统会创建ps和grep两个进程

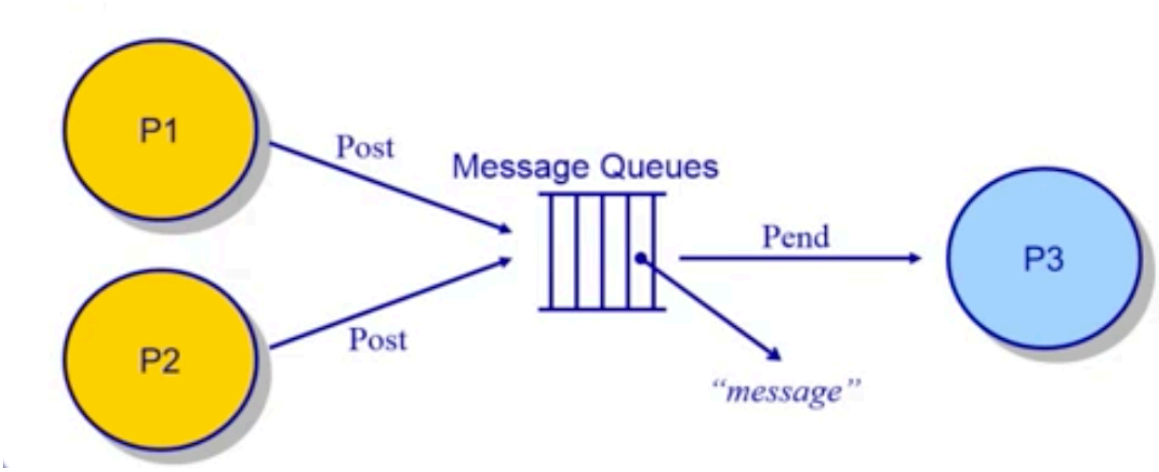
然后将ps进程的stdout(输出) 放到一个管道（内存中的一个buff）里面，然后grep从管道中stdin(输入)到grep进程

消息队列

也是进行数据交互，不需要是父子进程

消息队列按照FIFO(先进先出)来管理消息

- message： 作为一个字节序列存储
- message Queues：消息数组



共享内存

进程

- 每个进程都有私有地址空间
- 在每个地址空间内，明确的设置共享内存段

优点

快速，方便的共享数据

不足

必须同步数据访问

实现

由操作系统将同一块物理内存映射到两个进程不同的虚拟内存，即可共享内存