进程间通信

管道

在Linux命令中「|」就是一个管道（匿名管道），它的功能就是将前面的一个命令的输出作为后面一个命令的输入。
可以看出「管道传输数据是单向的」

「命名管道（FIFO先进先出）」需要使用 `mkfifo`创建
$ mkfifo pipeName(自定义)

管道这种通信方式效率很低，不适合进程间频繁的交换数据。（发送信息如果没有接收会阻塞直到管道数据被读完后才可以退出）

实现原理
- 匿名管道：对于匿名管道，它的通信范围是存在父子关系的进程。因为管道没有实体，也就是没有管道文件，只能通过 fork 来复制父进程 fd 文件描述符，来达到通信的目的
- 命名管道：对于命名管道，它可以在不相关的进程间也能相互通信。因为命令管道，提前创建了一个类型为管道的设备文件，在进程里只要使用这个设备文件，就可以相互通信

不管匿名命名进程写入的数据都是缓存在内核中，另一个进程读取的时候也在从内核读取，且遵循「先进先出」原则。管道传输的数据是无格式的流且大小受限！

消息队列

可以解决管道通信效率低的问题。
A进程给B进程发送消息，将消息放到消息队列后就可以正常返回，B进程需要的时候再去读取即可

消息队列是保存在内核中的消息链表，在发送数据时，会分成一个一个独立的数据单元（数据块），如果进程读取了这个消息内核就会将这个消息删除

消息队列生命周期随内核，如果没有释放消息或者关闭操作系统，消息会一直存在

缺点
- 通信不及时
- 大小限制：内核中定义了一个消息最大长度和消息队列的最大长度「MSGMAX」、「MSGMNB」
- 额外的数据拷贝开销：消息队列通信过程中，存在用户态与内核态之间的数据拷贝（写入从用户态拷贝到内核态，读取从内核态拷贝到用户态）

共享内存

解决了用户态和内核态之间消息拷贝的开销

共享内存的机制：拿出一块虚拟内存地址映射到相同物理内存中，省去拷贝环节，提高了通信速度

有Data race 风险

信号量

信号量解决内存共享data race风险

信号量是一个整型的计数器，用于实现进程间的互斥与同步，而不是用于缓存进程间通信数据

信号

异常情况下进程间通信需要使用「信号」通知进程

Ctrl+C产生「SIGINT」信号，表示终止该进程
Ctrl+Z产生「SIGTSTP」信号，表示停止该进程，但还未结束，挂起
kill -9 「SIGKILL」立即结束该进程
...

信号是进程间通信机制中「唯一的异步通信机制」。在任何时候发送任何信号给某一进程。
用户进程对信号的处理方式 有以下几种：
- 缺省：执行系统默认的操作
- 捕获：可以为专门的信号定义一个处理函数，当发生的时候执行相应的信号处理函数
- 忽略：当不希望处理某些信号，就可以忽略，不作任何处理

SIGKILL和SEEGSTOP是不可忽略捕获的

Socket

上面的所有的进程间通信都是基于同一台主机，想跨网络与不同主机上的进程之间通信，需要socket通信

系统调用「int socket(int domain, int type, int protocal)」

实现TCP字节流，socket的类型为「AF_INET 和 SOCK_STREAM」

实现UDP字节流，socket类型为「 AF_INET 和 SOCK_DGRAM」

实现本地进程间通信
- 字节流：「 AF_LOCAL 和 SOCK_STREAM」
- 数据报：「 AF_LOCAL 和 SOCK_DGRAM」
- 「AF_UNIX 和 AF_LOCAL 是等价的」
AF_UNIX 也属于本地 socket