1.通讯的基本角色

1.1指令流间通讯

不同的指令流代表了不同的活动。

指令流间的通信可以看作是不同活动之间的通信。

这些通信接口一般都是协程库和纤程库在用户模式实现的。

例如：企业中同一个员工的连个独立活动之间的信息传递

1.2线程间通讯

不同的线程代表了不同的时间预算。

线程间的通信可以看作是不同时间预算之间的通信。

这些通信接口一般是由操作系统在内核模式实现的。

例如：同一个企业不同部分的不同员工之间的信息传递

1.3进程间通讯

不同进程代表不同的地址空间。

进程间的通性可以看作是不同地址空间之间的通信。

这些通信接口智能由操作系统在内核模式实现的。

例如：同一个企业不同部分之间的信息传递。

1.4应用程序间通讯

不同应用程序代表不同的整体功能

应用程序间的通信可以看作是不同功能模块之间的通信

这些通信接口一般是由复杂的中间件结合IPC实现的

例如：不同企业之间的信息传递

对比：

通讯类别	沟通领域	开销	实现空间
指令流间	角色	极低	用户空间
线程间	时间	低	用户空间或内核空间
进程间	空间	高	内核空间
应用程序间	功能	极高	内核空间和用户空间中间件

问题：

（1）为什么指令流间和线程间通讯可以在用户空间实现，进程间通讯和应用程序间通讯只能在内核空间实现?

（2）在实践中，为什么线程间通讯总是在内核空间实现?

答案:

（1）进程和应用程序间通讯必然要跨越空间边界，而跨越空间边界只有操作系统才做得到。跨越时间边界和角色边界则可以在用户模式完成， 因为空间是共享的。

（2）线程是内核管理的，其阻塞式通信就必须在内核空间实现。指令流 则是用户空间管理的，其阻塞式通信可以在用户空间实现。

2.通讯的典型方式

2.1管道

管道——是一种简单的点对点通信工具，通常只能在两点之间进行先进先出的数据流传递。

在Linux中，管道分为有名和无名管道

无名管道——用于父子进程之间点对点通讯的工具。

父进程创建无名管道，然后调用fork()，父进程端使用一个端口，子进程端使用一个端口，并且只能父发子收，或者父收子发。

如果需要全双工，则只能创建两条无名管道

类别	Linux系统调用	描述
创建与销毁	pipe	创建一个无名管道，返回读写两端的文件描述符：如果父收子发则父端关闭[1]子端关闭[0]，如果父发子收则父端关闭[0]子端关闭[1]
	close	关闭无名管道的一端；当两端均关闭时，则无名管道自动销毁
读与写	read	从无名管道中读取数据
	write	向无名管道中写入数据

有名管道FIFO——用于任意进程之间通讯的工具。

任意进程均可创建有名管道，该管道将作为一个特殊文件存在。

此后，参与通信的进程可以通过打开并读写该管道文件进行通讯。

虽然有名管道允许都偶哥读者和写者，但是多个写者之间存在写交叉的风险。

2.2信号

信号

Linux系统中的一种回调函数机制，可以用来在不同进程之间或同一进程的不同线程之间发送轻量级通知。

当通知被送达时，线程暂停执行原程序，转去执行信号处理例程，然后再返回原程序继续执行。

类别	Linux系统调用	描述
登记信号处理例程	signal	为某个信号登记处理例程
	sigaction	为某个信号登记处理例程，相比signal可携带更多信息
发送信号	kill	发送信号到某个进程或线程
	sigqueue	发送信号到某个进程或线程，并携带少量数据

信号	编号	可自定义例程	默认动作	描述
SIGHUP	1	是	终止进程	终端挂起
SIGINT	2	是	终止进程	Ctrl+C
SIGKILL	9	否	终止进程	kill -9强制杀死进程
SIGSEGV	11	是	终止进程	段错误（非法内存访问）
SIGALRM	14	是	终止进程	alarm定时器到期
SIGTERM	15	是	终止进程	kill试图杀死进程
SIGUSR1/2	16/17	是	-	用户自定义信号1/2

2.3信号量与消息队列

信号量

Linux系统提供的标准信号量接口，同时具备资源计数和等待两种功能，非常适用于系统级编程中常见的生产者-消费者关系，或者复数资源的计数。

类别	Linux系统调用	描述
创建与销毁	sem_init	初始化信号量，可选择单进程内或进程间共享，还可指定初值
	sem_destroy	销毁信号量
获取	sem_wait	试图获取信号量，如果获取不成功则阻塞（阻塞式接口）
	sem_trywait	试图获取信号量，如果获取不成功则立即返回（非阻塞式接口）
	sem_timedwait	试图获取信号量，带超时（阻塞式接口）
释放	sem_post	释放信号量

消息队列

Linux提供的标准消息队列机制，可以再不同进程之间传送信息。

和管道不同，消息队列允许多个发送者和多个接收者，且收发均以消息位单位，无需考虑数据交叠的问题，

为了跨越地址空间，Linux会先将教习从发送者拷贝到内核，然后再从内核拷贝到接收者。

类别	Linux系统调用	描述
创建与销毁	msgget	创建并初始化消息队列
	msgctl	设置消息队列的属性，也可用来删除消息队列
发送	msgsnd	向消息队列发送（阻塞式或非阻塞式接口）
接受	msgrcv	从消息队列接收（阻塞式或非阻塞式接口）

SYSV/POSIX

上表列出的是System V消息队列接口，是Unix系统中最经典的消 息队列接口。Linux还提供另外一种mq_开头的POSIX消息队列接 口，提供同样的基本功能，但有更丰富的细节设置。

2.4线程迁移

线程迁移

线程迁移机制使线程在多个进程之间（以受控的方式）游走，在不同的执行阶段使用不同的地址空间和权限，但始终使用同一份CPU时间预算。迁移的接受者需要声明入口的地址和使用的 栈，并将进入入口的权能授权给合适的进程。被授权的进程内 的线程即可通过该入口进入接受者执行，执行完毕后返回自己 的原进程执行，就好像调用了一个函数那样。

这种机制一般存在于微内核中，在常规类Unix操作系统中是没有的。

同步与异步IPC

在同步IPC中，客户进程A向服务进程B发出请求后，客户进程A即阻塞，等待服务器进程B回复结果，也即数据的收发是一个操作。在异步IPC中，请求的发送和回复的接受是分立的两个操作。

线程迁移和其他IPC不同，它只能是同步的。其他IPC可以是异步的。

2.5共享内存

共享内存

Linux提供的地址空间共享机制，可以使两个地址空间之间共享一段物理内存（虚拟地址未必要一致）。此种方法等于是将裸内存直接暴露给了通信的参与者，需要参与者自行在共享内存 中组织合适的数据结构进行通信，内核不再介入数据传递。

类别	Linux系统调用	描述
创建与销毁	shmget	创建并初始化一块共享内存
	shmctl	设置共享内存的属性，也可用来删除共享内存
映射	shmat	将共享内存映射到当前进程
解除	shmdt	将共享内存从当前进程解除映射

2.6套接字及其他

该机制原本是网络通信准备的机制，当然也可以用于进程间通讯。

类别	系统调用	描述
套接字处理	socket socketpair	创建一个单独的或一对互相连接的套接字
	getsockopt setsockopt	读取或设置套接字设置
	getsockname getpeername	读取套接字或与其连接的套接字的地址
	bind	设置一个套接字的地址
	listen	监听对套接字的连接
	accept	接受对套接字的连接请求
	connect	向某套接字发起连接请求
	shutdown close	关闭套接字连接
数据收发处理	send/recvfrom send/recvmsg send/recvmmsg	接收或发送信息
多套接字等待	poll/select epoll	同时阻塞等待多个套接字，并在一个套接字返回数据时解除阻塞

消息总线

服务总线

消息队列的加强版，能够用于大型分布式系统中华部署于多台机器上的多台机器上的多个应用程序和或服务的总体通讯或协调。它集合了对象发现，同一编址，细节隐藏，消息投递和数据缓冲等功能。

接口定义语言IDL

一种程序接口描述规范，能够在不同的应用程序之间建立语言无关，系统无关和硬件无关的通信信道，从而使分布式系统中技术栈完全不同的程序能够相互协作

公共对象服务请求架构

Common Object Request Broker Architecture，CORBA

一种常见的多机分布式协作架构。能 够隐藏近乎一切软硬件差异，使分立的信息服务整合成一个系统。自1991年发布以来，历经超过十个大小版本更新，在各大中小公司业务中台搭建中均有一席之地。

IPC对比

IPC类别	跨线程	跨父子进程	跨任意进程	跨机器边界	跨服务组件
无名管道	可但没必要	可	不可	不可	不可
有名管道	可但没必要	可	可	不可	不可
信号	可	可	可	不可	不可
信号量	可	可	可	不可	不可
消息队列	可	可	可	不可	不可
线程迁移	不可	可	可	不可	不可
共享内存	可但没必要	可	可	有RDMA则可	可但很麻烦
套接字	可但没必要	可	可	可	可但较麻烦
消息总线	可但没必要	可但没必要	可但没必要	可	可