在Linux操作系统中实现内部进程通信 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E5_9C_A8Linux_E6_93_c103_144424.htm Linux给我们提供了丰富的内部进程通信机制，包括共享内存、内存映射文件、先入先出（FIFO）、接口（sockets）以及多种用于同步的标识。在本文中，我们主要讨论一下共享内存和内存映射文件技术。 一般来说，内部进程通信（interprocess communication）也就是IPC，是指两个或两个以上进程以及两个或者两个以上线程之间进行通信联系。每个IPC机制都有不同的强项或者弱点，不过没有一个IPC机制包含内建的同步方法。因此程序员不但需要自己在程序中实现同步，而且还需要为了利用IPC机制而自己开发通信协议。 共享内存 使用共享内存和使用malloc来分配内存区域很相似。使用共享内存的方法是： 1.对一个进程/线程使用shmget分配内存区域。 2.使用shmat放置一个或多个进程/线程在共享内存中，你也可以用shmctl来获取信息或者控制共享区域。 3.使用shmdt从共享区域中分离。 4.使用shmctl解除分配空间 下面是个例子: //建立共享内存区域 intshared_id. char *region. const intshm_size = 1024. shared_id = shmget(IPC_PRIVATE,//保证使用唯一ID shm_size, IPC_CREAT | IPC_EXCL |//创建一个新的内存区域 S_IRUSR | S_IWUSR).//使当前用户可以读写这个区域 //交叉进程或生成进程. //将新建的内存区域放入进程/线程 region = (char*) shmat(segment_id, 0, 0). //其他程序代码 ... //将各个进程/线程分离出来 shmdt(region). //破坏掉共享内存区域 shmctl(shared_id, IPC_RMID, 0). 共享内存是Linux中最快速的IPC方法。他也是

一个双向过程，共享区域内的任何进程都可以读写内存。这个机制的不利方面是其同步和协议都不受程序员控制，你必须确保将句柄传递给了子进程和线程。