UC Day12

预习课

预习内容

内存壁垒

内存壁垒

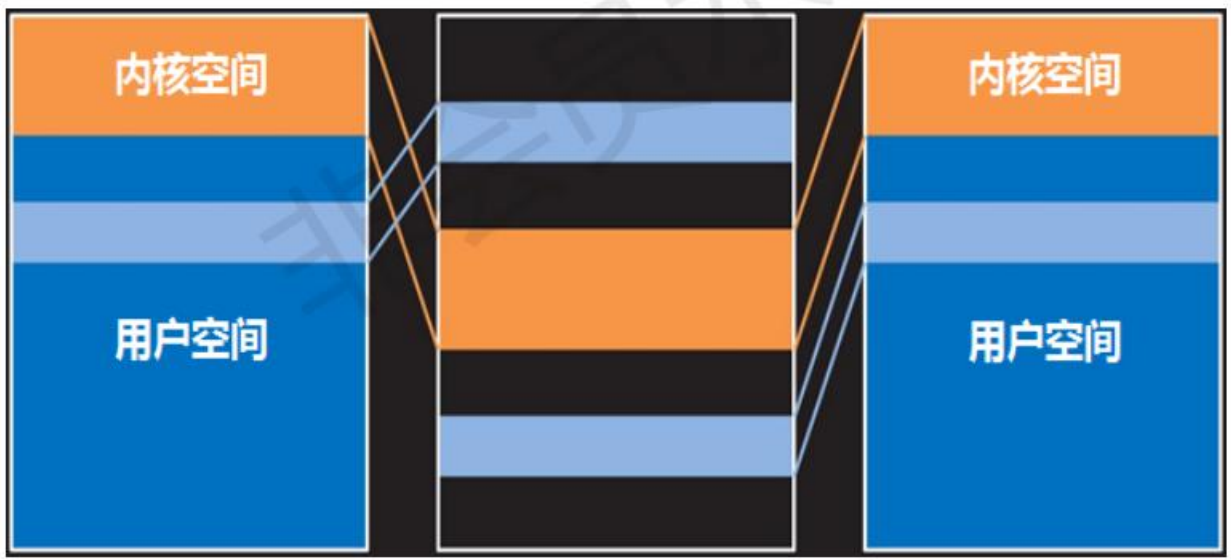
内存壁垒

知识讲解

- 每个进程的用户空间都是0~3G-1，但它们所对应的物理内存却是各自独立的，系统为每个进程的用户空间维护一张专属于该进程的内存映射表，记录虚拟内存到物理内存的对应关系，因此在不同进程之间交换虚拟内存地址是毫无意义的
- 所有进程的内核空间都是3G~4G-1，它们所对应的物理内存只有一份，系统为所有进程的内核空间维护一张内存映射表init_mm.pgd，记录虚拟内存到物理内存的对应关系，因此不同进程通过系统调用所访问的内核代码和数据是同一份

内存壁垒

- 用户空间的内存映射表会随着进程的切换而切换，内核空间的内存映射表则无需随着进程的切换而切换



内存壁垒

知识讲解

- 正如前文所述，Unix/Linux系统中的每个进程都拥有4G字节大小专属于自己的虚拟内存空间，除去内核空间的1G以外，每个进程都有一张独立的内存映射表(又名内存分页表)记录着虚拟内存页和物理内存页之间的映射关系
- 同一个虚拟内存地址，在不同的进程中，会被映射到完全不同的物理内存区域，因此在多个进程之间以交换虚拟内存地址的方式交换数据是不可能的
- 鉴于进程之间天然存在的内存壁垒，要想实现多个进程间的数据交换，就必须提供一种专门的机制，这就是所谓的进程间通信(InterProcess Communication, IPC)

直播课见

UC

C/C++教学体系

目录

进程间通信的种类

进程间通信的种类

知识讲解

- 命令行参数
  - 在通过exec函数创建新进程时，可以为其指定命令行参数，借助命令行参数可以将创建者进程的某些数据传入新进程
  - execl ("login", "login", "username", "password", NULL);
- 环境变量
  - 类似地，也可以在调用exec函数时为新进程提供环境变量
  - sprintf (envp[0], "USERNAME=%s", username);
  - sprintf (envp[1], "PASSWORD=%s", password);
  - execle ("login", "login", NULL, envp);
- 内存映射文件
  - 通信双方分别将自己的一段虚拟内存映射到同一个文件中

进程间通信的种类

知识讲解

- 管道
  - 管道是Unix系统中最古老的进程间通信方式，并且所有的Unix系统和包括Linux系统在内的各种类Unix系统也都提供这种进程间通信机制。管道有两种限制
  - 管道都是半双工的，数据只能沿着一个方向流动
  - 管道只能在具有公共祖先的进程之间使用。通常一个管道由一个进程创建，然后该进程通过fork函数创建子进程，父子进程之间通过管道交换数据
  - 大多数Unix/Linux系统除了提供传统意义上的无名管道以外，还提供有名管道，对后者而言第二种限制已不复存在

进程间通信的种类

知识讲解

- 共享内存
  - 共享内存允许两个或两个以上的进程共享同一块给定的内存区域。因为数据不需要在通信诸方之间来回复制，所以这是速度最快的一种进程间通信方式
- 消息队列
  - 消息队列是由系统内核负责维护并可在多个进程之间共享存取的消息链表。它的优点是：传输可靠、流量受控、面向有结构的记录、支持按类型过滤
- 信号量
  - 与共享内存和消息队列不同，信号量并不是为了解决进程间的数据交换问题。它所关注的是有限的资源如何在无限的用户间合理分配，即资源竞争问题
- 本地套接字
  - BSD版本的有名管道。编程模型和网络通信统一

有名管道

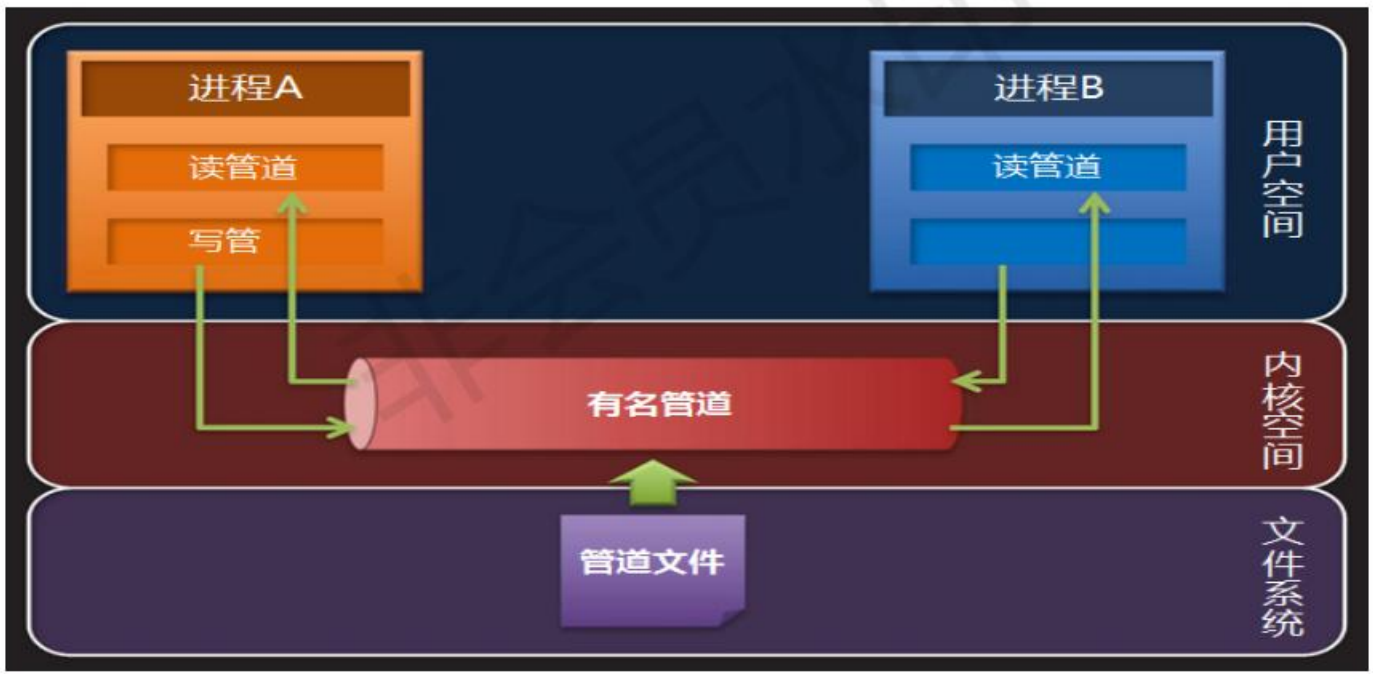
有名管道

知识讲解

- 有名管道亦称FIFO，是一种特殊的文件，它的路径名存在于文件系统中。通过mkfifo命令可以创建管道文件
  - $ mkfifo myfifo
- 即使是毫无亲缘关系的进程，也可以通过管道文件通信
  - $ echo 'Hello, FIFO !' > myfifo
  - $ cat myfifo
  - Hello, FIFO !
- 管道文件在磁盘上只有i节点没有数据块，也不保存数据

有名管道

- 基于有名管道实现进程间通信的逻辑模型



知识讲解

有名管道

- 有名管道不仅可以用于Shell命令，也可以在代码中使用基于有名管道实现进程间通信的编程模型

步骤	进程A	函数	进程B	步骤
1	创建管道	mkfifo	——	——
2	打开管道	open	打开管道	1
3	读写管道	read/write	读写管道	2
4	关闭管道	close	关闭管道	3
5	删除管道	unlink	——	——

知识讲解

有名管道

知识讲解

- #include <sys/stat.h>
- int mkfifo(char const* pathname, mode_t mode);
  - 功能：创建有名管道
  - 参数：pathname：有名管道名，即管道文件的路径。
    mode：权限模式。
  - 返回值：成功返回0，失败返回-1

无名管道

无名管道

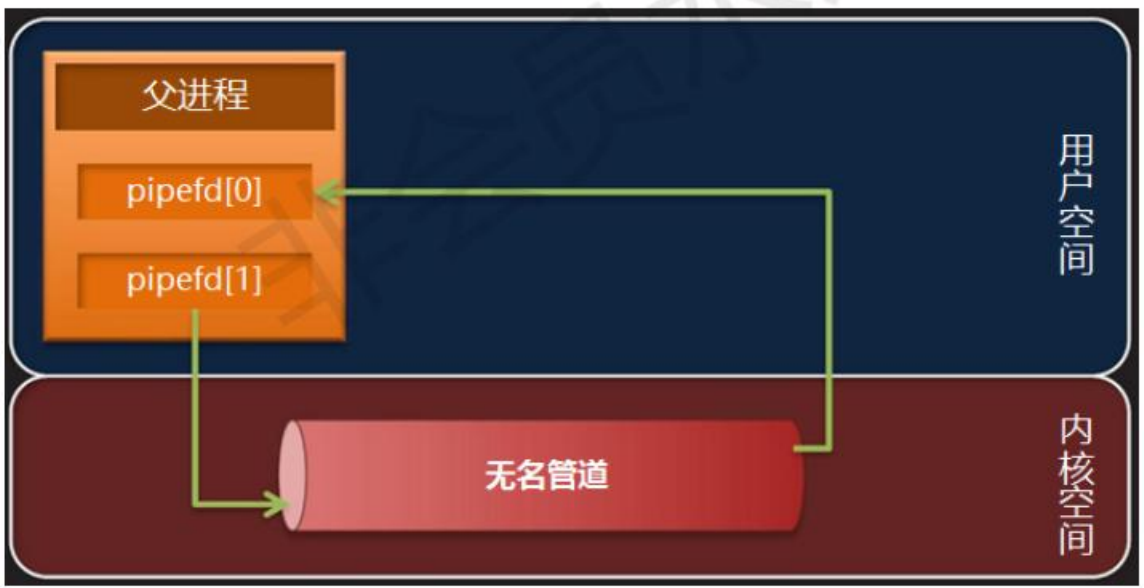
知识讲解

- 无名管道是一个与文件系统无关的内核对象，主要用于父子进程之间的通信，需要用专门的系统调用函数创建
- #include <unistd.h>
- int pipe(int pipefd[2]);
  - 功能：创建无名管道
  - 参数：pipefd 输出两个文件描述符：
    pipefd[0] - 用于从无名管道中读取数据；
    pipefd[1] - 用于向无名管道中写入数据。
  - 返回值：成功返回0，失败返回-1

无名管道

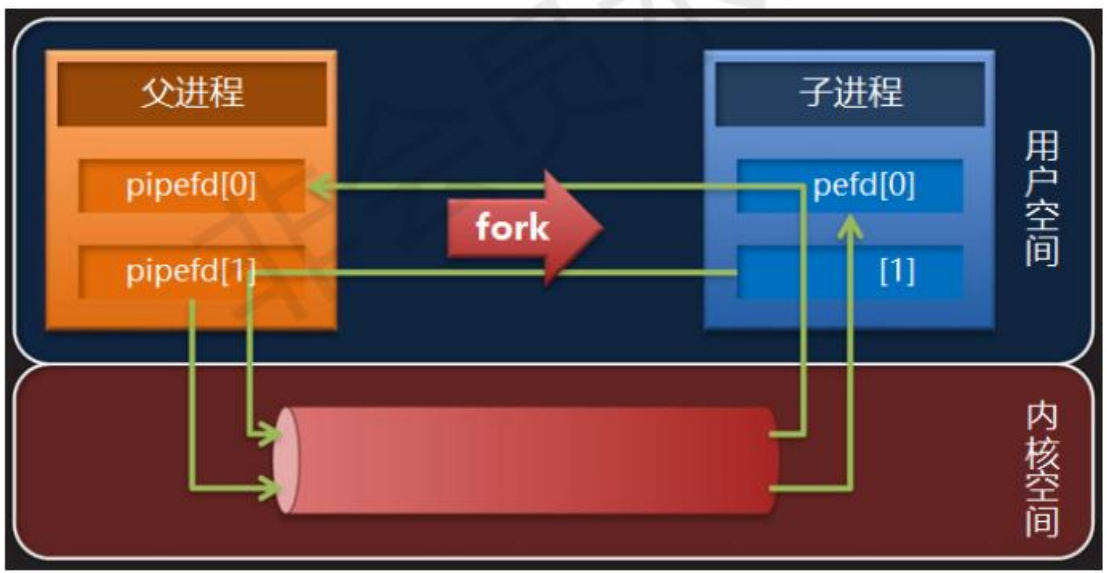
知识讲解

- 基于无名管道实现进程间通信的编程模型
  - 1. 父进程调用pipe函数在系统内核中创建无名管道对象，并通过该函数的输出参数pipefd，获得分别用于读写该管道的两个文件描述符pipefd[0]和pipefd[1]



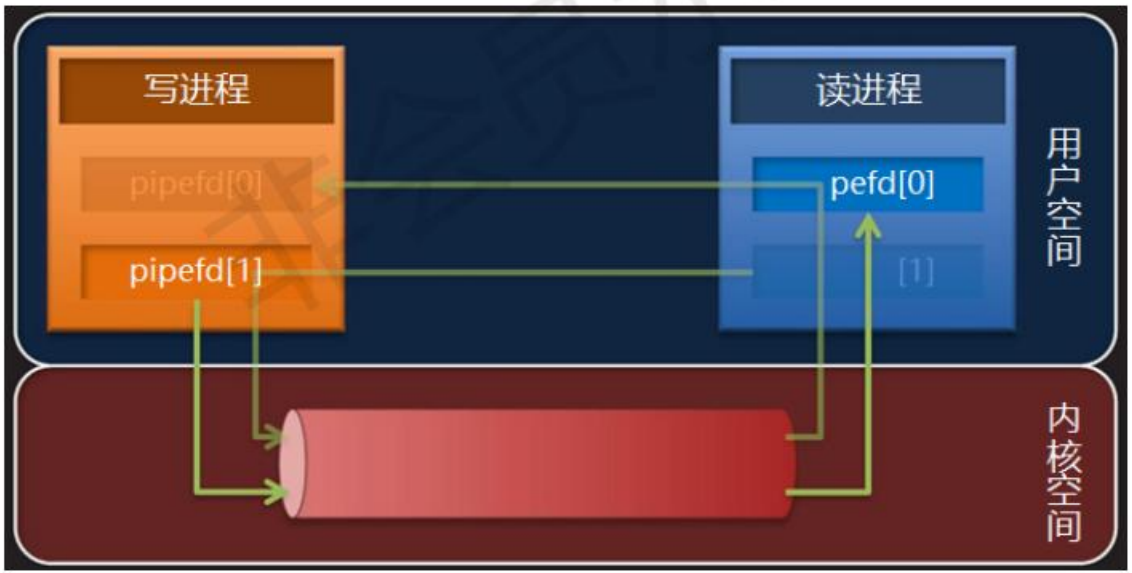
无名管道

- 基于无名管道实现进程间通信的编程模型
  - 2.父进程调用fork函数，创建子进程。子进程复制父进程的文件描述符表，因此子进程同样持有分别用于读写该管道的两个文件描述符pipefd[0]和pipefd[1]



无名管道

- 基于无名管道实现进程间通信的编程模型
  - 3. 负责写数据的进程关闭无名管道对象的读端文件描述符pipefd[0]，而负责读数据的进程则关闭该管道的写端文件描述符pipefd[1]



无名管道

知识讲解

- 基于无名管道实现进程间通信的编程模型
  - 4. 父子进程通过无名管道对象以半双工的方式传输数据。如果需要在父子进程间实现双向通信，较一般化的做法是创建两个管道，一个从父流向子，一个从子流向父

无名管道

知识讲解

- 基于无名管道实现进程间通信的编程模型
  - 5. 父子进程分别关闭自己所持有的写端或读端文件描述符。在与一个无名管道对象相关联的所有文件描述符都被关闭以后，该无名管道对象即从系统内核中被销毁

特殊情况

特殊情况

知识讲解

- 1、从写端已被关闭的管道读取，只要管道中还有数据，依然可以被正常读取，一直到管道中没有数据了，这时read函数会返回0(既不是返回-1，也不是阻塞)，指示读到文件尾。
- 2、向读端已被关闭的管道写入会直接触发SIGPIPE(13)信号。该信号的默认操作时终止执行写入动作的进程。但如果执行写入动作的进程事先已将对SIGPIPE(13)信号的处理设置为忽略或捕获，则write函数会返回-1，并置errno为EPIPE。

特殊情况

知识讲解

- 3、系统内核通常为每个管道维护一个大小为4096字节的内存缓冲区。如果写管道时发现缓冲区的空闲空间不足以容纳此次write所要写入的字节，则write会阻塞，直到缓冲区的空闲空间变得足够大为止。
- 4、读取一个写端处于开放状态的空管道，将直接导致read函数阻塞。

谢谢

UC Day012

复习课

管道符号

管道符号

知识讲解

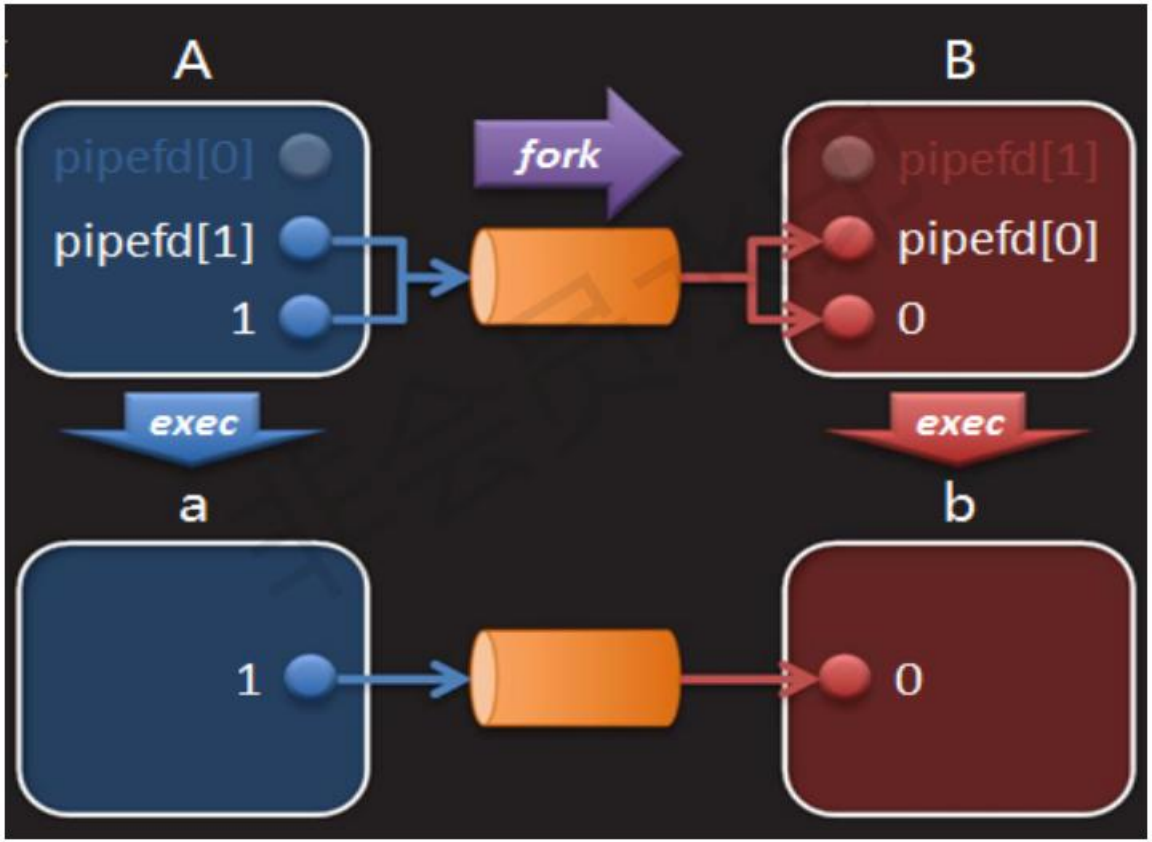
- Unix/Linux系统中的多数Shell环境都支持通过管道符号“|”将前一个命令的输出作为后一个命令的输入的用法
  - $ ls -l /etc | more
  - $ ifconfig | grep inet
- 系统管理员经常使用这种方法，把多个简单的命令连接成一条工具链，去解决一些通常看来可能很复杂的问题
  - 命令1 | 命令2 | 命令3

管道符号

知识讲解

- 假设用户输入如下命令：a | b
  - Shell进程调用fork函数创建子进程A
  - 子进程A调用pipe函数创建无名管道，而后执行
    - dup2 (pipefd[1],STDOUT_FILENO);
  - 子进程A调用fork函数创建子进程B，子进程B执行
    - dup2 (pipefd[0],STDIN_FILENO);
  - 子进程A和子进程B分别调用exec函数创建a、b进程
  - a进程所有的输出都通过写端进入管道，而b进程所有的输入则统统来自该管道的读端。这就是管道符号的工作原理

管道符号



下节课见