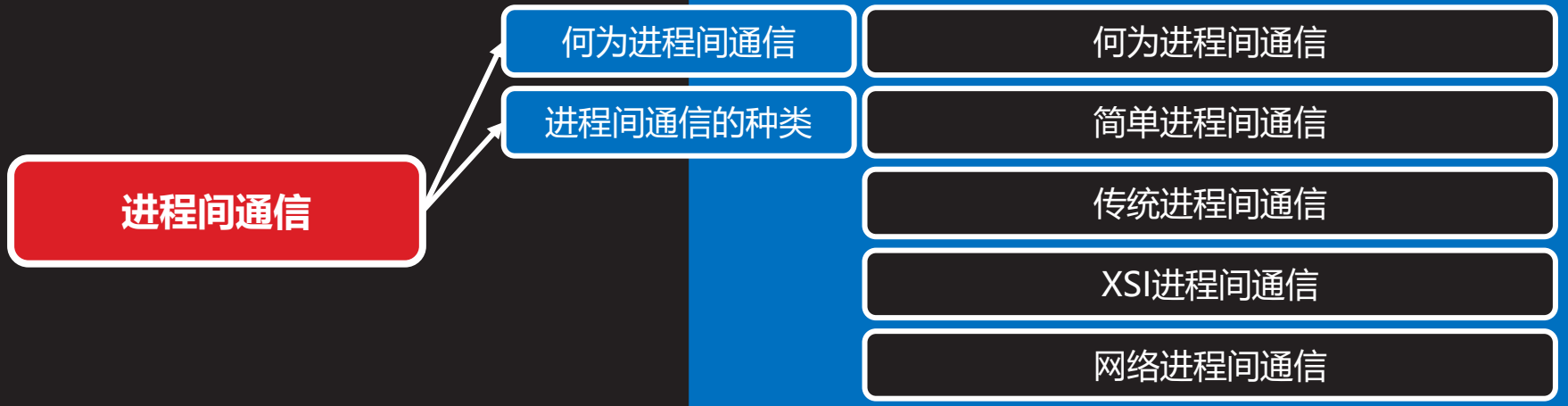
Technology Tarena 达内科技

Unix系统高级编程

进程间通信和管道 Unit21

进程间通信


进程间通信
何为进程间通信
进程间通信的种类
何为进程间通信
简单进程间通信
传统进程间通信
XSI进程间通信
网络进程间通信

Technology Tarena 达内科技

何为进程间通信

何为进程间通信

- 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)

Technology Tarena 达内科技

进程间通信的种类

简单进程间通信

- 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
- 信号
  - SIGUSR1/SIGUSR2，sigaction/sigqueue/附加数据

知识讲解

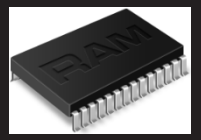
传统进程间通信

- 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
  - 基于SVR4的管道事实上是全双工的，即数据可在一根管道中双向流动。但是POSIX.1只提供半双工管道，出于移植性的考虑，不妨依然坚持“管道都是半双工的”这一假定

知识讲解

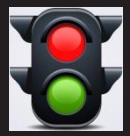
XSI进程间通信

- 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



知识讲解

网络进程间通信

- 套接字
  - 脱胎于System V的共享内存、消息队列和信号量，它们最大的问题就是都没有使用文件描述符。因此也就无法对它们使用多路I/O复用——select和poll。更不能象访问文件那样，访问系统内核中的IPC对象。从本质上讲这是对Unix系统向来秉承的一切皆文件思想的一种无耻的背叛
  - 从这个意义上讲，源自BSD的网络编程接口——套接字——恰恰体现了一种对Unix固有文化的虔诚的皈依，因为它完全是基于文件描述符的。所有对文件起作用的系统调用，无论是基本的读写、还是对内核对象的控制，甚或是多路I/O复用，都可以无一例外地应用于网络或本机中的不同进程。这足以证明BSD把一切皆文件进行到底的决心和斗志

知识讲解

管道

Technology Tarena 达内科技

有名管道

有名管道

- 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
```

- 在文件系统中，管道文件被显示成这个样子

```
prw-rw-r-- 1 tarena tarena 0 12月  5 10:07 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
```

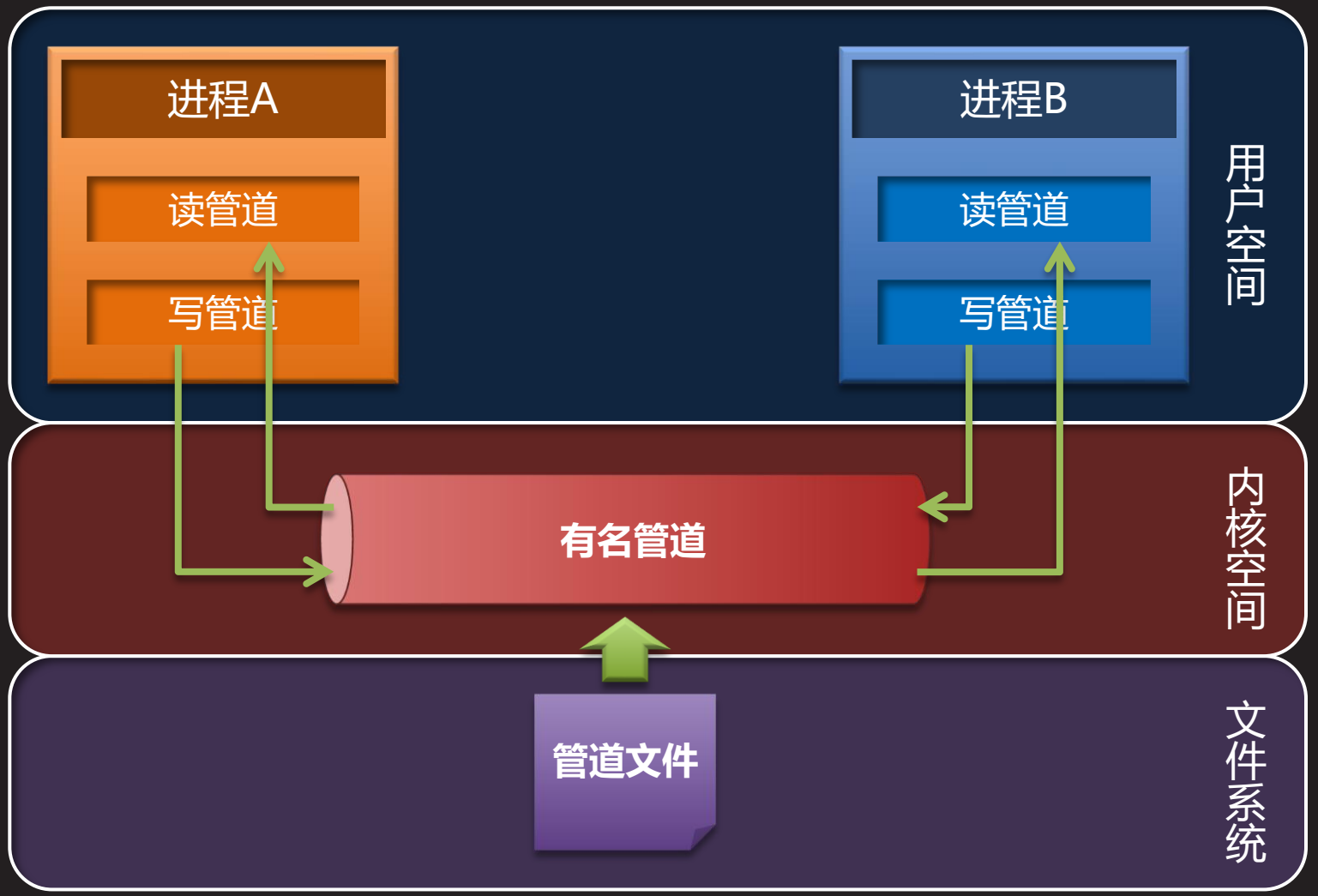
```
Hello, FIFO !
```

- 管道文件在磁盘上只有i节点没有数据块，也不保存数据

知识讲解

有名管道（续1）

- 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（续2）

- 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	——	——

- 其中除了mkfifo函数是专门针对有名管道的，其它函数都与操作普通文件没有任何差别
- 有名管道是文件系统的一部分，如不删除，将一直存在

知识讲解

有名管道（续3）

- 创建有名管道文件

```
#include <sys/stat.h>

int mkfifo (const char* pathname, mode_t mode);
```

成功返回0，失败返回-1

  - pathname：文件路径
  - mode：权限模式
- 例如
  -

```
if (mkfifo ("myfifo", 0666) == -1) {
    perror ("mkfifo");
    exit (EXIT_FAILURE);
}
```

知识讲解

基于有名管道的进程间通信

【参见：wfifo.c、rfifo.c】

课堂练习

- 基于有名管道的进程间通信

Technology Tarena 达内科技

无名管道

无名管道

- 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]);
```

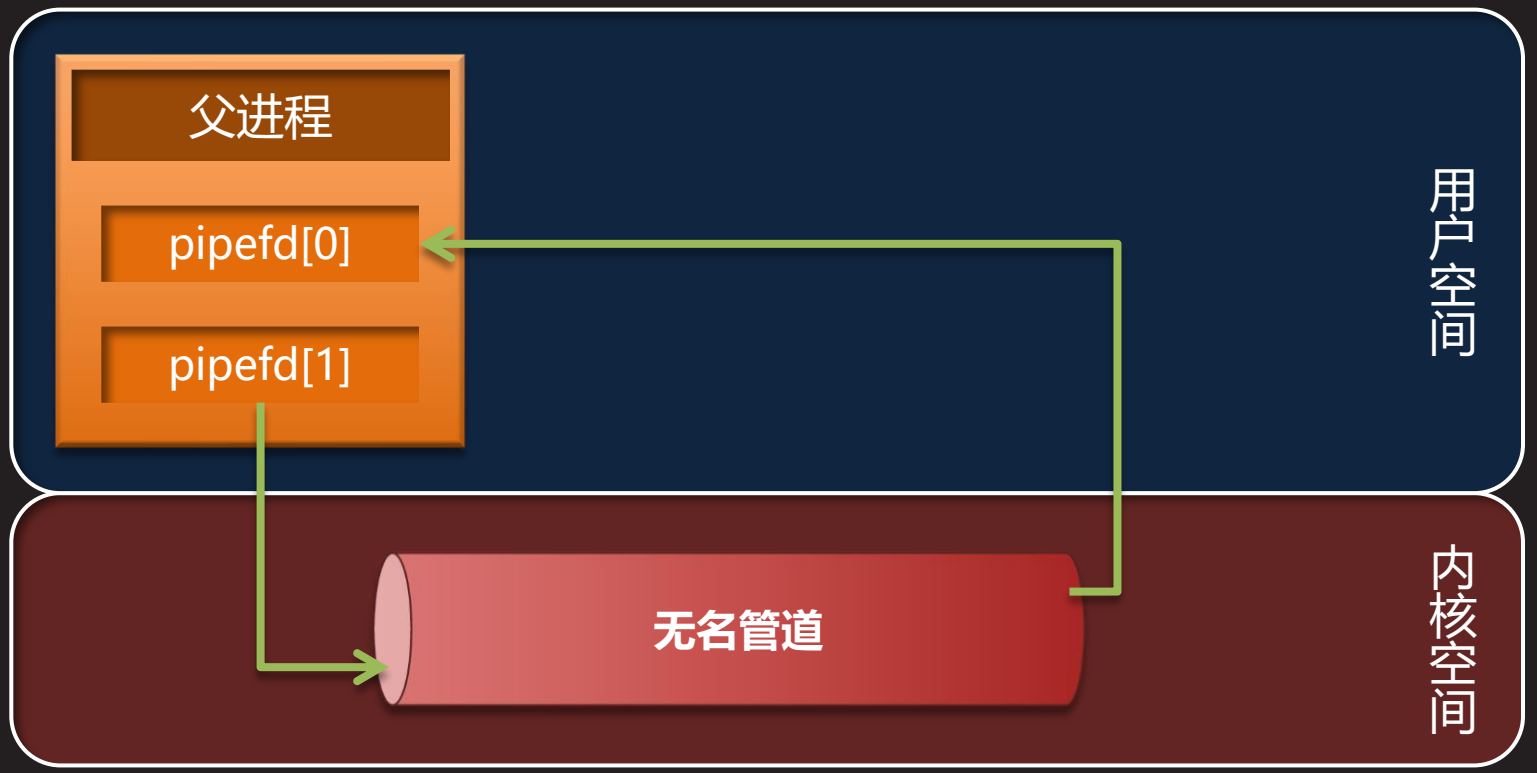
成功返回0，失败返回-1

  - ***pipefd***：输出两个文件描述符，***pipefd***[0]用于从所创建的无名管道中读取数据，***pipefd***[1]用于向该管道写入数据

- 例如
  - int pipefd[2];
    if (pipe (pipefd) == -1) {
        perror ("pipe"); exit (EXIT_FAILURE); }

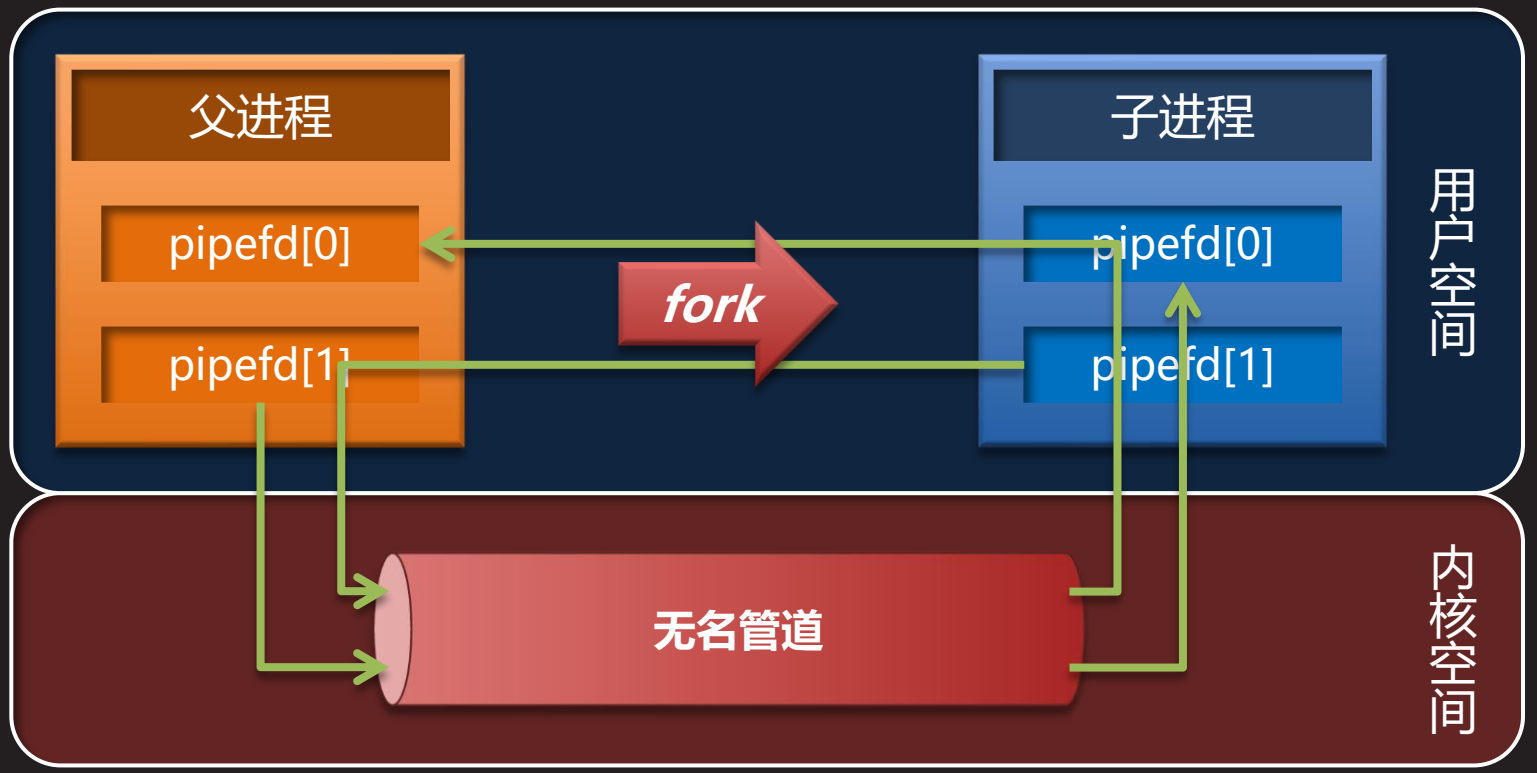
无名管道（续1）

- 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（续2）

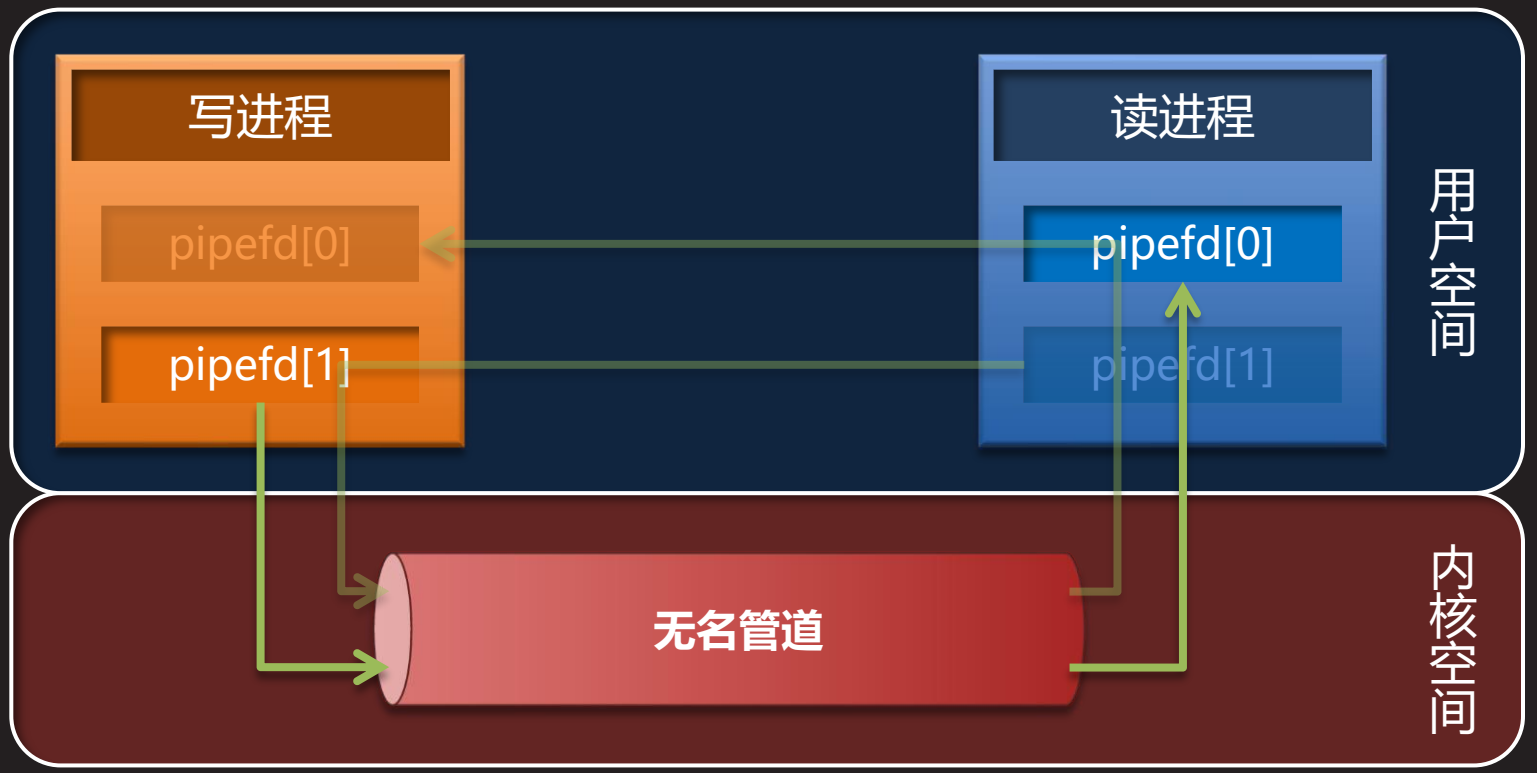
- 基于无名管道实现进程间通信的编程模型
  2. 父进程调用fork函数，创建子进程。子进程复制父进程的文件描述符表，因此子进程同样持有分别用于读写该管道的两个文件描述符***pipefd***[0]和***pipefd***[1]



知识讲解

无名管道（续3）

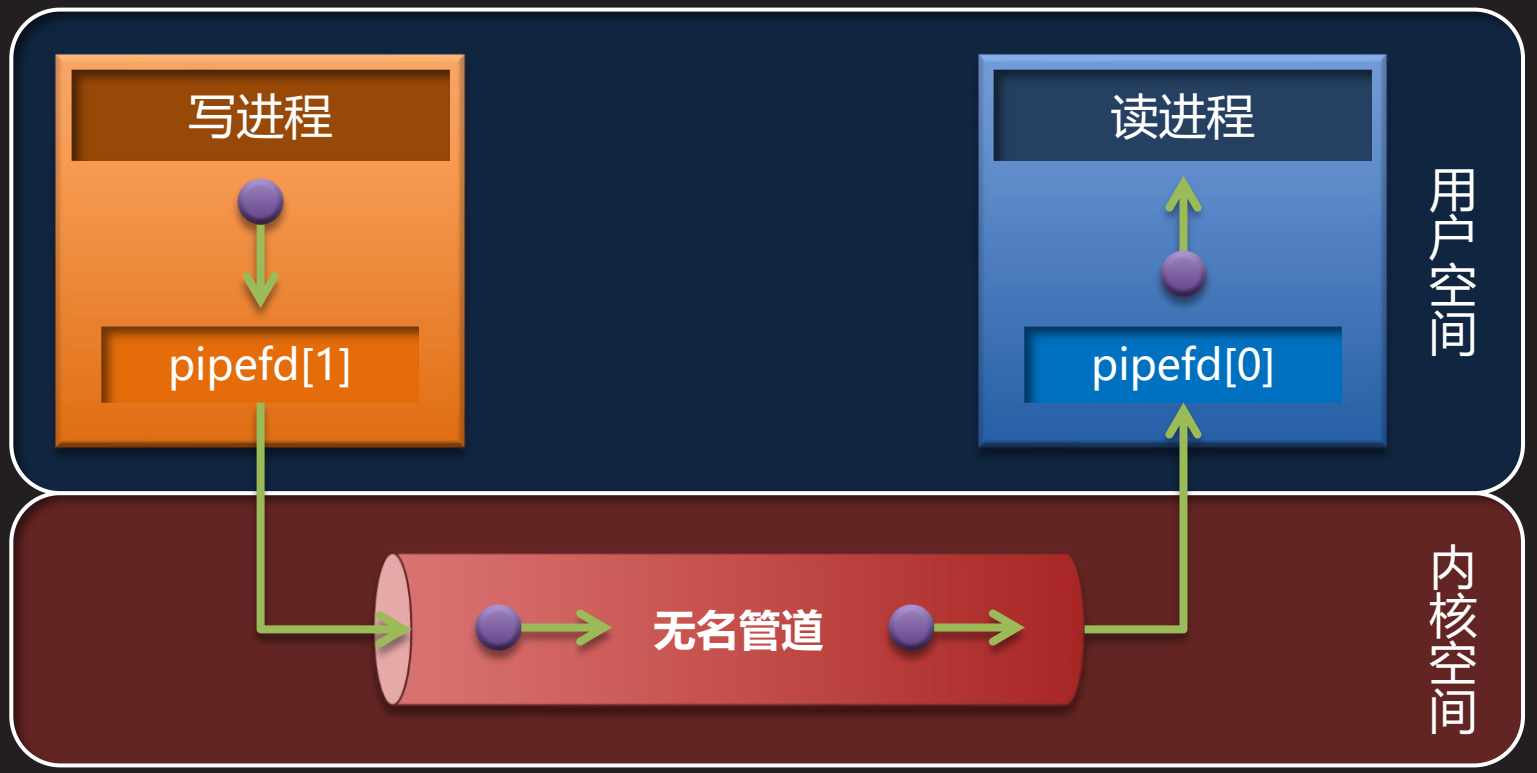
- 基于无名管道实现进程间通信的编程模型
  3. 负责写数据的进程关闭无名管道对象的读端文件描述符***pipefd***[0]，而负责读数据的进程则关闭该管道的写端文件描述符***pipefd***[1]



知识讲解

无名管道（续4）

- 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



知识讲解

无名管道（续5）

- 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

知识讲解

基于无名管道的进程间通信

【参见：pipe.c】

课堂练习

- 基于无名管道的进程间通信

Technology Tarena 达内科技

管道符号

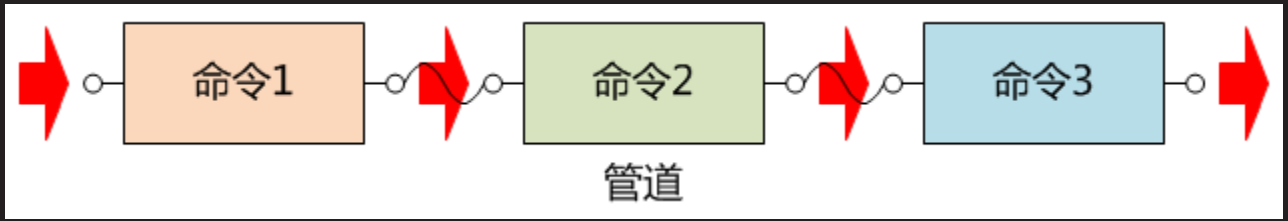
管道符号

- 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
```

```
总用量 1260
-rw-r--r--  1 root   root     2076  4月  3  2012 bash.bashrc
-rw-r--r--  1 root   root    58753  3月 31  2012 bash_completion
-rw-r--r--  1 root   root      356  4月 20  2012 bindresvport.blacklist
lrwxrwxrwx  1 root   root       15  6月 18 09:21 blkid.tab -> /dev/.blkid.tab
...
--更多--
```

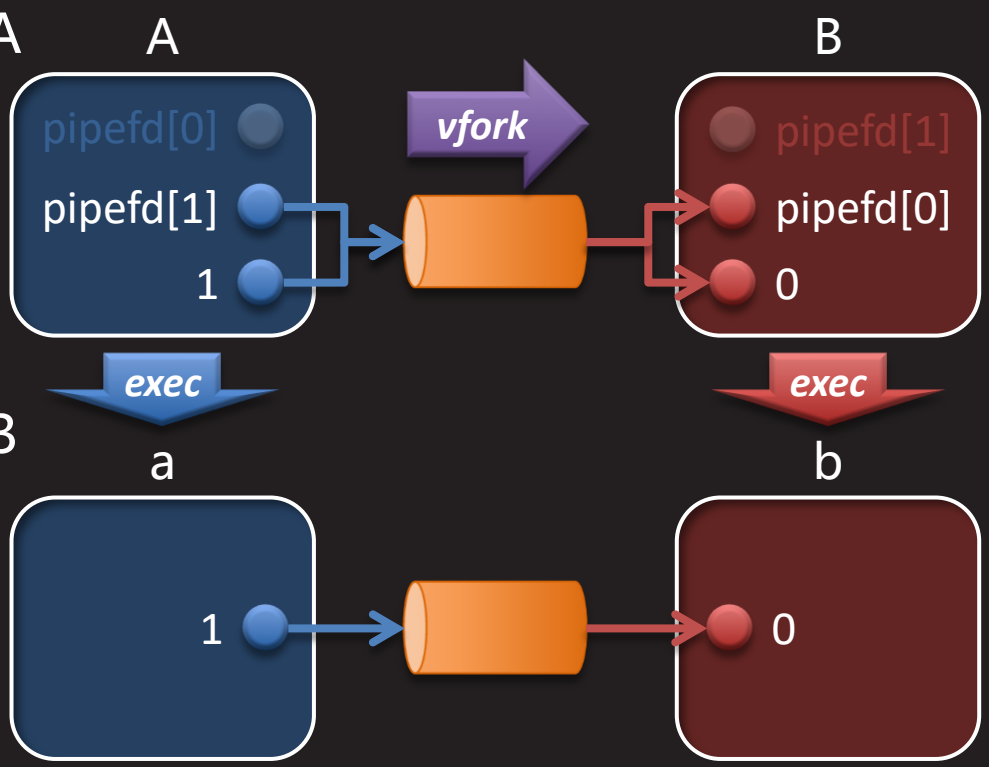
- 系统管理员经常使用这种方法，把多个简单的命令连接成一条工具链，去解决一些通常看来可能很复杂的问题



知识讲解

管道符号（续1）

- 假设用户输入如下命令：a | b
  - 调用pipe函数创建无名管道
  - 调用vfork函数创建子进程A
    - 调用dup2函数，将管道写端复制于标出输出
      dup2 (pipefd[1], STDOUT_FILENO);
    - 调用exec函数创建a进程
  - 调用vfork函数创建子进程B
    - 调用dup2函数，将管道读端复制于标准输入
      dup2 (pipefd[0], STDIN_FILENO);
    - 调用exec函数创建b进程
  - 等待并回收所有子进程，获取其终止状态



知识讲解

管道符号的原理性实现

【参见：out.c、in.c、shell.c】

课堂练习

- 管道符号的原理性实现

Technology Tarena 达内科技

特殊情况

特殊情况

- 无论是有名管道，还是无名管道，在对它们进行读写操作的过程中，都难免会遇到一些特殊情况
- 从写端已关闭的管道读取
  - 只要管道中还有数据，依然可以正常读取，一直读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
- 向读端已关闭的管道写入
  - 会直接触发SIGPIPE(13)信号。该信号的默认操作是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者捕获并从对该信号的处理函数中返回，则write函数会返回-1，并置errno为EPIPE

特殊情况（续1）

- 系统内核会为每个管道维护一个内存缓冲区，缓冲区的大小通常为4096字节，在<limits.h>头文件中被定义为PIPE_BUF宏
  - 如果写管道时发现缓冲区中的空闲空间不足以容纳此次write所要写入的字节，则write函数会阻塞，直到缓冲区中的空闲空间变得足够大为止
  - 如果同时有多个进程向同一个管道写入数据，而每次调用write函数写入的字节数都不大于PIPE_BUF，则这些write操作不会相互穿插，反之，如果单次写入的字节数了超过PIPE_BUF，则它们的write操作可能会相互穿插
  - 读取一个缓冲区为空的管道，将直接导致read函数阻塞
  - 若管道是非阻塞的，则上述所有的阻塞都会立即返回失败

知识讲解

Technology Tarena 达内科技

总结和答疑