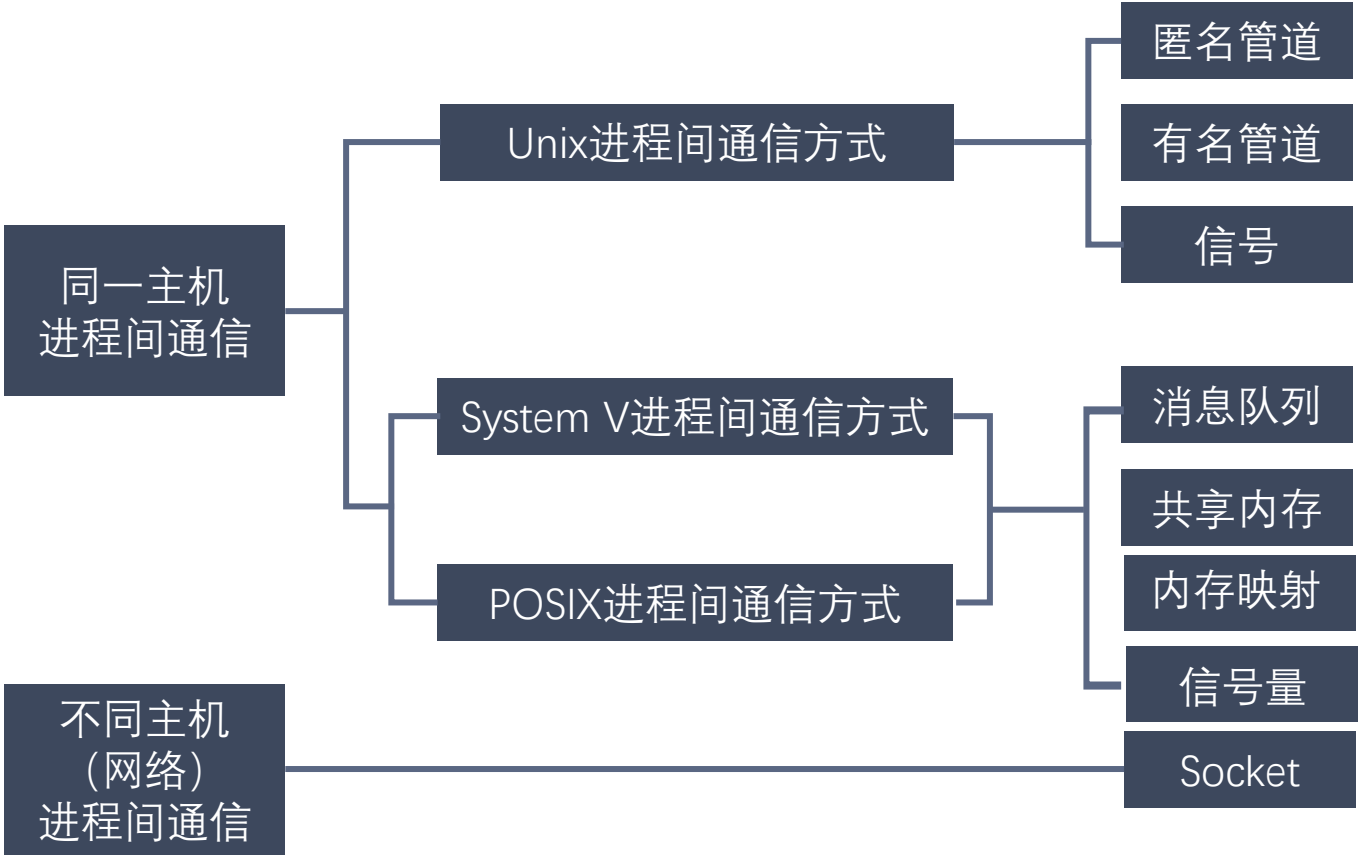
进程间通信

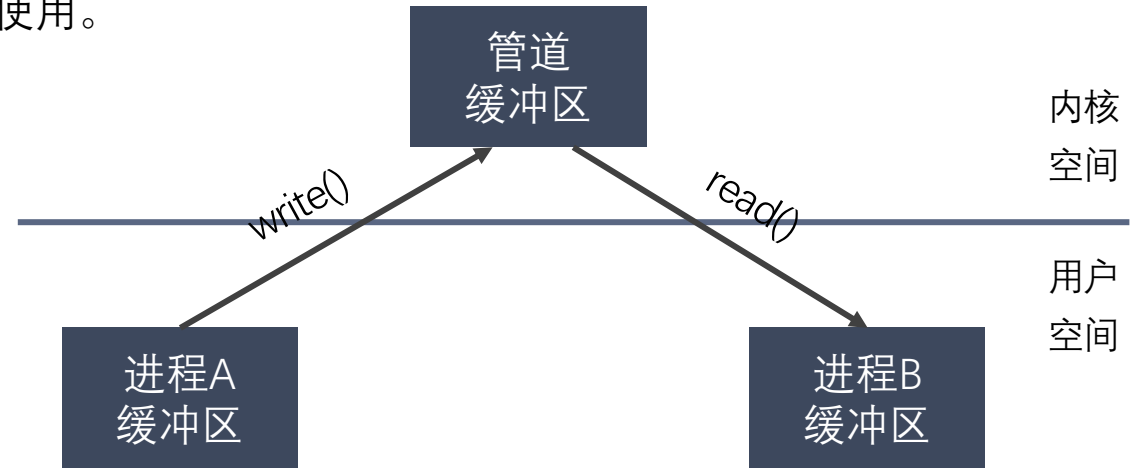
- 进程是一个独立的资源分配单元，不同进程（这里所说的进程通常指的是用户进程）之间的资源是独立的，没有关联，不能在一个进程中直接访问另一个进程的资源。
- 但是，进程不是孤立的，不同的进程需要进行信息的交互和状态的传递等，因此需要进程间通信（IPC：Inter Processes Communication）。
- 进程间通信的目的：
  - 数据传输：一个进程需要将它的数据发送给另一个进程。
  - 通知事件：一个进程需要向另一个或一组进程发送消息，通知它（它们）发生了某种事件（如进程终止时要通知父进程）。
  - 资源共享：多个进程之间共享同样的资源。为了做到这一点，需要内核提供互斥和同步机制。
  - 进程控制：有些进程希望完全控制另一个进程的执行（如 Debug 进程），此时控制进程希望能够拦截另一个进程的所有陷入和异常，并能够及时知道它的状态改变。


同一主机
进程间通信
Unix进程间通信方式
匿名管道
有名管道
信号
System V进程间通信方式
消息队列
共享内存
POSIX进程间通信方式
内存映射
信号量
不同主机
（网络）
进程间通信
Socket

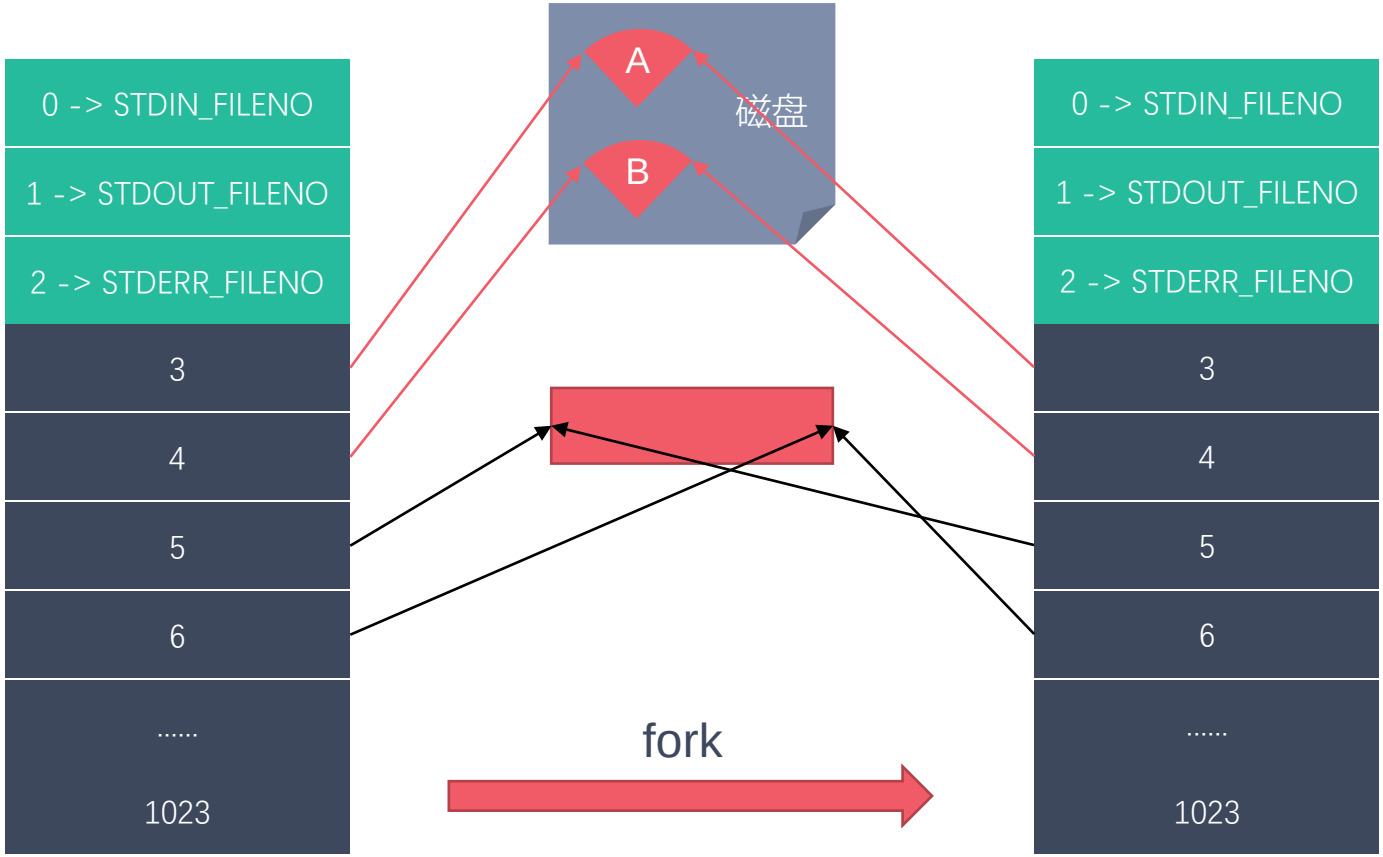
- 管道也叫无名（匿名）管道，它是是 UNIX 系统 IPC（进程间通信）的最古老形式，所有的 UNIX 系统都支持这种通信机制。
- 统计一个目录中文件的数目命令：ls | wc -l，为了执行该命令，shell 创建了两个进程来分别执行 ls 和 wc。

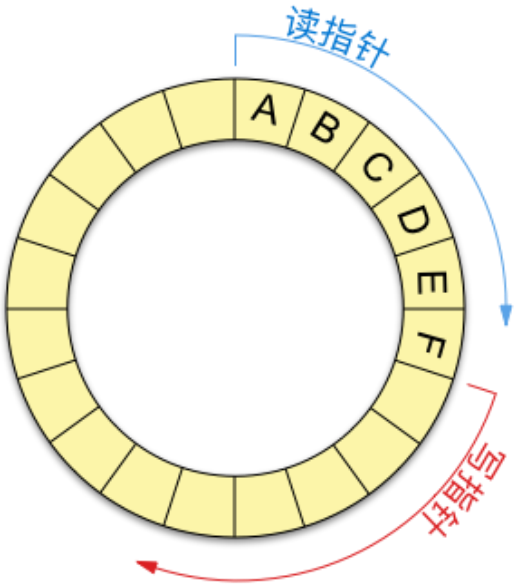
- 管道其实是一个在内核内存中维护的缓冲器，这个缓冲器的存储能力是有限的，不同的操作系统大小不一定相同。
- 管道拥有文件的特质：读操作、写操作，匿名管道没有文件实体，有名管道有文件实体，但不存储数据。可以按照操作文件的方式对管道进行操作。
- 一个管道是一个字节流，使用管道时不存在消息或者消息边界的概念，从管道读取数据的进程可以读取任意大小的数据块，而不管写入进程写入管道的数据块的大小是多少。
- 通过管道传递的数据是顺序的，从管道中读取出来的字节的顺序和它们被写入管道的顺序是完全一样的。

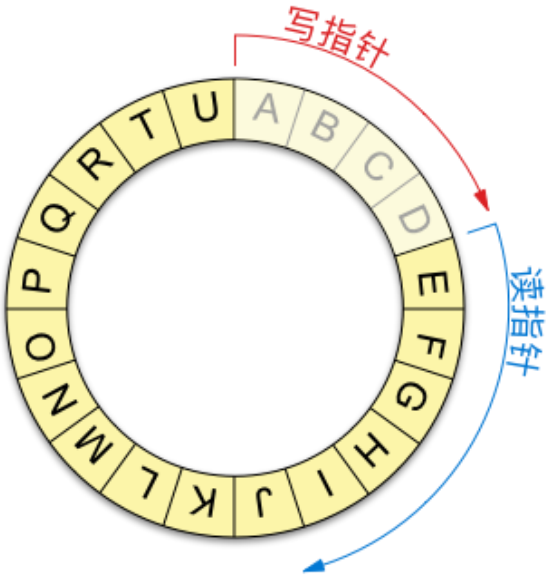
- 在管道中的数据的传递方向是单向的，一端用于写入，一端用于读取，管道是半双工的。
- 从管道读数据是一次性操作，数据一旦被读走，它就从管道中被抛弃，释放空间以便写更多的数据，在管道中无法使用 `lseek()` 来随机的访问数据。
- 匿名管道只能在具有公共祖先的进程（父进程与子进程，或者两个兄弟进程，具有亲缘关系）之间使用。



05 / 为什么可以使用管道进行进程间通信







- 创建匿名管道

```
#include <unistd.h>

int pipe(int pipefd[2]);
```

- 查看管道缓冲大小命令

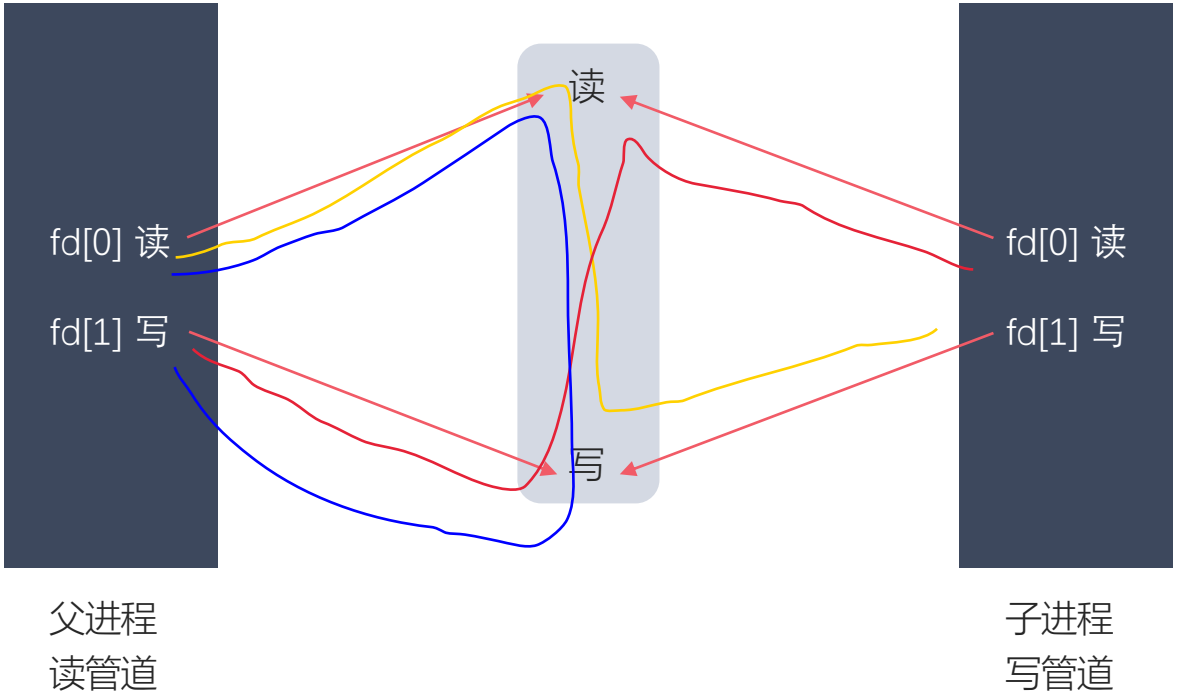
```
ulimit -a
```

- 查看管道缓冲大小函数

```
#include <unistd.h>

long fpathconf(int fd, int name);
```

07 / 匿名管道的使用



- 匿名管道，由于没有名字，只能用于亲缘关系的进程间通信。为了克服这个缺点，提出了有名管道（FIFO），也叫命名管道、FIFO文件。
- 有名管道（FIFO）不同于匿名管道之处在于它提供了一个路径名与之关联，以 FIFO 的文件形式存在于文件系统中，并且其打开方式与打开一个普通文件是一样的，这样即使与 FIFO 的创建进程不存在亲缘关系的进程，只要可以访问该路径，就能够彼此通过 FIFO 相互通信，因此，通过 FIFO 不相关的进程也能交换数据。
- 一旦打开了 FIFO，就能在它上面使用与操作匿名管道和其他文件的系统调用一样的I/O系统调用了（如read()、write()和close()）。与管道一样，FIFO 也有一个写入端和读取端，并且从管道中读取数据的顺序与写入的顺序是一样的。FIFO 的名称也由此而来：先入先出。

- 有名管道（FIFO）和匿名管道（pipe）有一些特点是相同的，不一样的地方在于：

  1．FIFO 在文件系统中作为一个特殊文件存在，但 FIFO 中的内容却存放在内存中。

  2．当使用 FIFO 的进程退出后，FIFO 文件将继续保存在文件系统中以便以后使用。

  3．FIFO 有名字，不相关的进程可以通过打开有名管道进行通信。

- 通过命令创建有名管道

  mkfifo 名字

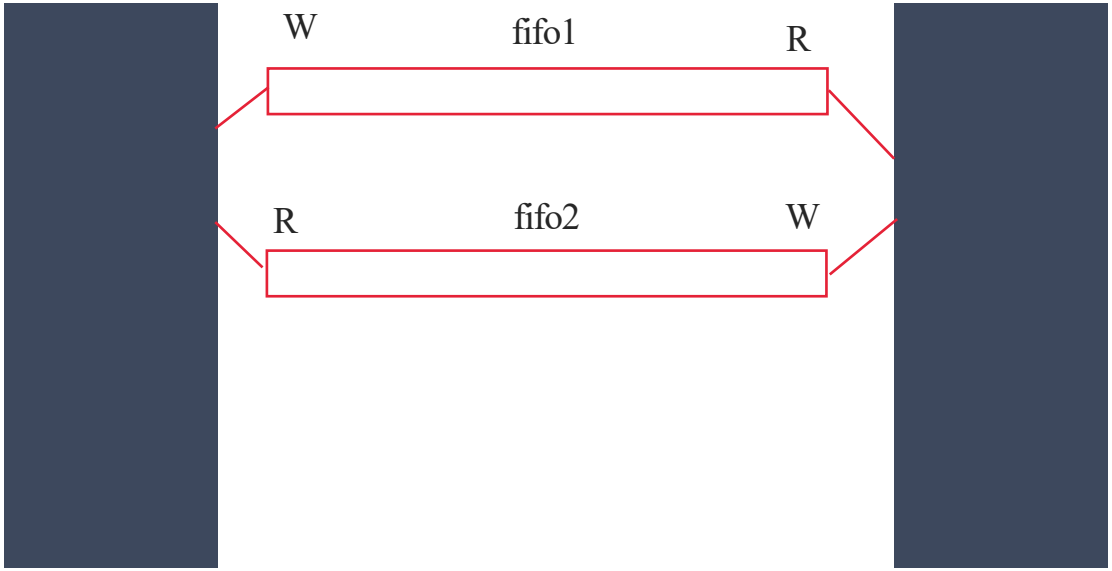
- 通过函数创建有名管道

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *pathname, mode_t mode);
```

- 一旦使用 mkfifo 创建了一个 FIFO，就可以使用 open 打开它，常见的文件 I/O 函数都可用于 fifo。如：close、read、write、unlink 等。
- FIFO 严格遵循先进先出（First in First out），对管道及 FIFO 的读总是从开始处返回数据，对它们的写则把数据添加到末尾。它们不支持诸如 lseek() 等文件定位操作。

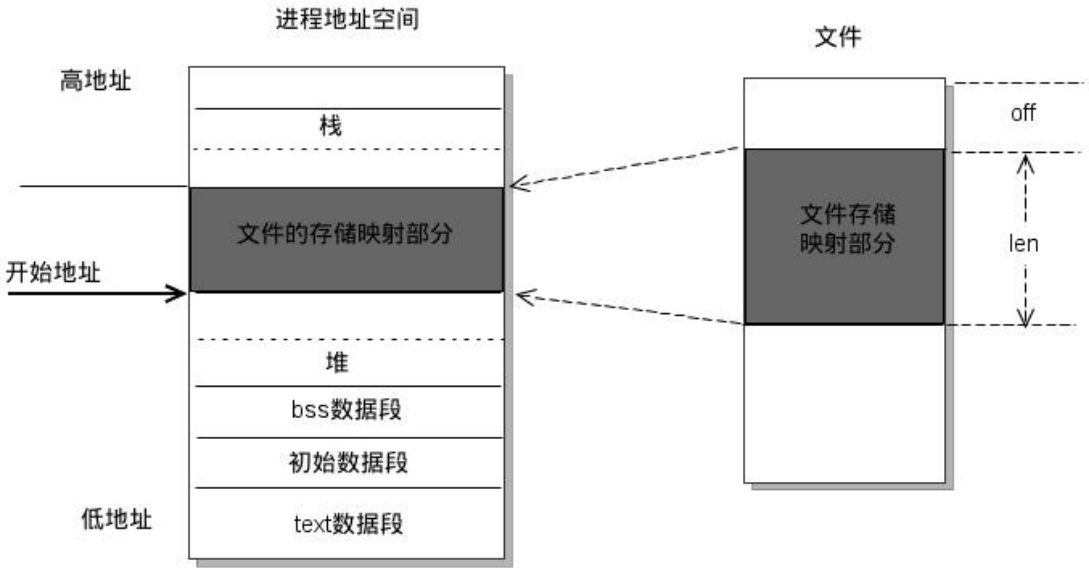
使用有名管道完成聊天的功能

进程A
1.以只写的方式打开管道1
2.以只读的方式打开管道2
3.循环的写读
while(1)
{
  获取键盘录入fgets
  写管道1
  读管道2
}



进程B
1.以只读的方式打开管道1
2.以只写的方式打开管道2
3.循环的读写
while(1)
{
  读管道1
  获取键盘录入fgets
  写管道2
}

- 内存映射（Memory-mapped I/O）是将磁盘文件的数据映射到内存，用户通过修改内存就能修改磁盘文件。



- `#include <sys/mman.h>`
- `void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);`
- `int munmap(void *addr, size_t length);`

- 如果对mmap的返回值(ptr)做++操作(ptr++)，munmap是否能够成功?
- 如果open时O_RDONLY，mmap时prot参数指定PROT_READ | PROT_WRITE会怎样?
- 如果文件偏移量为1000会怎样?
- mmap什么情况下会调用失败?
- 可以open的时候O_CREAT一个新文件来创建映射区吗?
- mmap后关闭文件描述符，对mmap映射有没有影响?
- 对ptr越界操作会怎样?

牛客大学
-专业求职辅导-

THANKS

关注【牛客大学】公众号
回复“牛客大学”获取更多求职资料