


第4章 管道与重定向

——匿名管道

主 讲：黄 茹

- 一条管道命令中至少涉及两个进程，以ls -l|more为例：ls命令的输出通过一条管道连接more命令的输入
- 要解决两个问题：

  1.创建一条管道

  2.对管道前后连接的两条命令进程进行I/O重定向

创建匿名管道——pipe

可以使用系统调用pipe创建匿名管道:

pipe	
功能	创建一条匿名管道
头文件	/user/include/unistd.h
函数原型	int pipe(int parr[2]);
参数	parr 存放管道两端文件描述符的数组
返回值	-1　发生错误
	0　成功

匿名管道的说明

匿名管道的内部实现隐藏在内核中，实质是一个以队列方式读写的内核缓冲区。可以使用系统调用pipe创建管道:

调用pipe创建管道后，管道两端和两个文件描述符相关联，记录在parr数组中。parr[0]中存放管道的读端，parr[1]中存放的是管道的写端

示例程序1

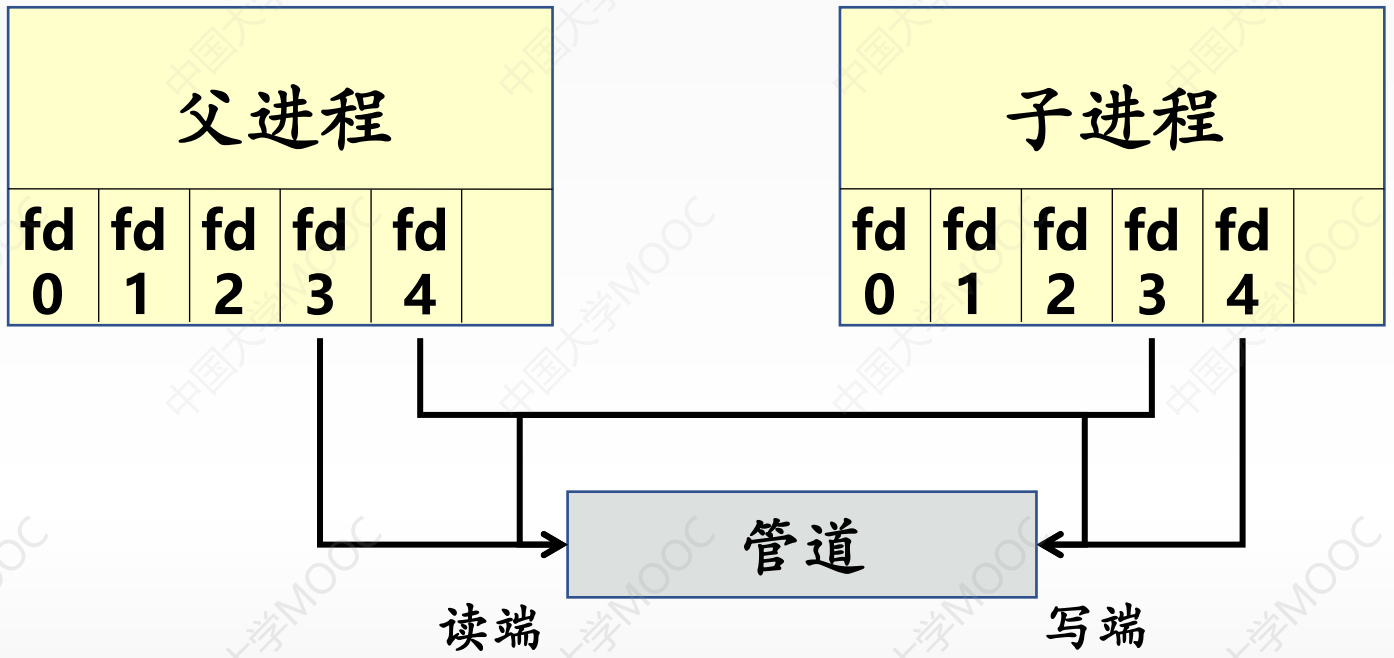
```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

main()  {
   int pfd[2];  char buf[81];
   if(pipe(pfd)==-1)  {perror("pipe");       exit(1);   }
   printf("The pipe will read from %d, write to %d.\n", pfd[0], pfd[1]);
   write(pfd[1],"This is write to pipe!\n",23);
   read(pfd[0],buf,23);
   printf("%s",buf);
}
```

只涉及到一个进程和管道的关系

父子进程使用匿名管道通信

- 匿名管道以文件描述符的形式供进程使用
- 调用fork后，子进程会复制父进程打开的文件描述符列表
- 结论：调用fork前调用pipe



示例程序2


```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

main(){
  int pid,pfd[2]; char buf[80];
  if(pipe(pfd)==-1)        {perror("pipe"); exit(1);  }
  pid=fork();
  if(pid<0)        {perror("fork");  exit(1);  }
  else if(pid==0){
        close(pfd[1]);
        if(read(pfd[0],buf,80)>0)
              printf("Message from child:%s\n",buf);
        close(pfd[0]);
        exit(0);
  }
  else   {
        close(pfd[0]);
        if(write(pfd[1],"Pipe is a useful tool!",23)!=-1)
              printf("Parent is writing the message!\n");
        close(pfd[1]);
        wait(NULL);
        exit(0);
  }
}
```

延伸思考：

- 问题：
  - 如果子进程同时也要发送信息给父进程该怎么办?
  - 没有亲缘关系的进程可以使用匿名管道通信么?
  - 父子进程的运行顺序无法预测，是否每次运行都会是父进程先写，子进程再读?

使用管道的注意事项

- 执行读操作时
  - 如果管道中无数据，则读进程将被挂起直到数据被写进管道
  - 如果所有写进程都关闭了管道的写端时，read返回0，意味着文件的结束。

使用管道的注意事项

- 执行写操作时
  - 当管道已满时，写进程再对管道做写操作时，写进程会被阻塞
  - 如果所有读进程关闭了管道的读端，再执行写操作时，写进程将会收到SIGPIPE信号，若该信号不能终止进程，则write调用返回-1，并将errno置为EPIPE
  - 写入管道的数据大小，POSIX规定内核不会拆分小于512字节的块，因此如果有两个进程向管道写数据，只要每一个进程都限制消息不大于512字节，写入的消息就不会被内核拆分。

谢谢大家！