Pipe

张海宁

贵州大学

hnzhang1@gzu.edu.cn

May 30, 2018

Overview

Pipe

What is a Pipe

Pipe

We use the term *pipe* to mean connecting a data flow from one process to another.

shell command

```
cat <<"EOF" | grep "abc"
```

$$\begin{tikzcd} \boxed{\text{stdin}} \arrow[r] & \boxed{\text{cat «"EOF"}} \arrow[r] & \boxed{\text{piple}} \arrow[r] & \boxed{\text{grep "abc"}} \arrow[r] & \boxed{\text{stdout}} \end{tikzcd}$$

进程管道

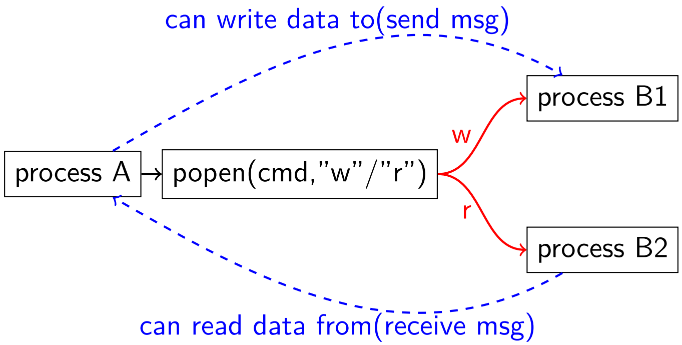
Process pipe

Perhaps the simplest way of passing data between two programs is with the *popen* and *pclose* functions.

原型

```
#include <stdio.h>
FILE *popen(const char *command, const char *open_mode);
int pclose(FILE *stream_to_close);
```

popen


can write data to(send msg)
process B1
w
process A
popen(cmd,"w"/"r")
r
process B2
can read data from(receive msg)

read data from child process

12-pipeRead.c

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main(){
 FILE * f; char buf[1000]; int len;
 memset(buf,'\0',1000);
 f=popen("uname -a","r"); sleep(5);
 if(f==NULL){
  perror("error in create another process!"); exit(-1); }
 len = fread(buf,sizeof(char),1000,f);
 if(len>0){
  printf("the out put of uname -a is:\n%s\n",buf); }
 printf("finished!"); pclose(f); exit(0);
}
```

read data from child process

13-pipeRead.c

```
./pipe &
[2] 5254
[1]   Done                    ./pipe
$ ps -j
USER          PID  PPID  PGID   SESS JOBC STAT COMMAND
hainingzhang  1583  1582  1583     0    1 S     -bash
hainingzhang  5254  1583  5254     0    1 S     ./pipe
hainingzhang  5255  5254  5254     0    1 Z    (uname)
$ the out put of uname -a is:
Darwin HainingdeMacBook-Pro.local 17.5.0 Darwin Kernel
 Version 17.5.0: Mon Mar  5 22:24:32 PST 2018;
  root:xnu-4570.51.1~1/RELEASE_X86_64 x86_64

finished!
```

send data to child process

13-pipeWrite.c

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main(){
 FILE * f; char buf[1000]; int len;
 memset(buf,'\0',1000);
 sprintf(buf,"I can say a, b ,c and d.");
 f=popen("wc -w","w");
 if(f==NULL){
  perror("error in create another process!");
  exit(-1); }
 fwrite(buf,sizeof(char),strlen(buf),f);
 printf("finished!"); pclose(f); exit(0);
}
```

popen 的特性

1. 启动一个新进程时会先启动一个 shell(回顾一下 system 函数)
2. cmd 是作为一个参数传递给 shell 来运行的

pipe 调用

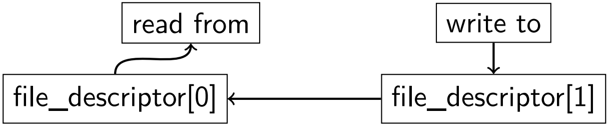
pipe 调用

pipe 调用也可以在两个程序间传递数据，与 popen 不同的是，pipe 不需要启动 shell，同时还提供了对读写数据的更多控制。

pipe 原型

```
#include <unistd.h>
int  pipe(int fildes[2]);
```

成功返回 0，否则返回-1 。



pipe 例子

13-pipe.c

```
int main(){
 int fd[2]; pid_t cpid; char buf[100];
 char * data="data to be sent."; int len;
 memset(buf,'\0',sizeof(buf));
 if(pipe(fd)==0){
  cpid=fork();
  if(cpid==-1){ perror("fork error!"); exit(-1); }
 if(cpid==0){
  len=read(fd[0],buf,100);
  printf("read %d bytes, it is: %s.\n",len,buf); exit(0);
 }else{
   len=write(fd[1],data,strlen(data));
   printf("wrote %d bytes to another process.\n",len);
  } } exit(0); }
```

命名管道 FIFO

FIFO 文件

我们涉及到管道的两种实现方式都只是在有相互关联的程序之间传递数据。如果想在没有关联的程序之间传递数据的话，可以使用 FIFO 文件，通常也被称为命名管道（name pipe）。
命名管道是一种特殊的文件，其行为和我们之前的管道是类似的。

命名管道的使用

1. 创建命名管道

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

2. 打开命名管道
   使用 open 系统调用
   - 读
     使用 read
   - 写
     使用 write

创建命名管道并写数据

13-fifoWrt.c

```
int main(){
 mkfifo("/tmp/myfifo",0777);
 int fd = open("/tmp/myfifo",O_WRONLY);
 int i=0; char c = 'a';
 for(i=0;i<20;i++){
  write(fd,&c,1);
  c++;
  sleep(1);
 }
 close(fd);
 exit(0);
}
```

通过命名管道读数据

13-fifoRd.c

```
int main(){
 int fd = open("/tmp/myfifo",O_RDONLY);
 char buf;
 int len=read(fd,&buf,1);
 while(len>0){
  printf("the fifo content is:%c\n",buf);
  len=read(fd,&buf,1);
 }
 close(fd);
 exit(0);
}
```

作业

编写一个客户端和一个服务端程序，要求：

1. 实现双向通信
2. 使用命名管道技术

The End

Appendix

本课程相关资源下载

1. ppt
   https://github.com/gmsft/ppt/tree/master/linux
2. 实验指导书
   https://github.com/gmsft/ppt/tree/master/book/linux

about man page

The manual is generally split into eight numbered sections, organized as follows (on Research Unix, BSD, macOS and Linux):

section	**description**
1	General commands
2	System calls
3	Library function(C standard library)
4	Special files(devices) and drivers
5	File formats and conventions
6	Games and screensavers
7	Miscellanea
8	System administration commands and daemons

Table: man page

在终端中运行 man read 与 man 2 read ，观察其输出的区别。