**2.2 进程的管道通信**

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

#include<stdlib.h>

#include<sys/wait.h>

#include <sys/types.h>

#include<string.h>

int pid1,pid2; // 定义两个进程变量

int main( ) {

int fd[2],a;

char OutPipe[100],InPipe[4100]; // 定义两个字符数组

char c1='1',c2='2';

pipe(fd); // 创建管道

while((pid1 = fork( )) == -1); // 如果进程1创建不成功,则空循环

if(pid1 == 0) { // 如果子进程1创建成功,pid1为进程号

lockf(fd[1],1,0); // 锁定管道

sprintf(OutPipe,"\n Child process 1 is sending message!\n"); // 给Outpipe赋值

write(fd[1],OutPipe,strlen(OutPipe));

for(int i=0;i<2000;i++)

write(fd[1],&c1,sizeof(c1));

sleep(5);

lockf(fd[1],0,0); // 解除管道的锁定

exit(0); // 结束进程1

}

else {

while((pid2 = fork()) == -1); // 若进程2创建不成功,则空循环

if(pid2 == 0) {

lockf(fd[1],1,0);

sprintf(OutPipe,"\n Child process 2 is sending message!\n");

write(fd[1],OutPipe,strlen(OutPipe));

for(int i=0;i<2000;i++)

write(fd[1],&c2,1); // 向管道写入数据

sleep(5);

lockf(fd[1],0,0);

exit(0);

}

else {

memset(InPipe,'\0',sizeof(InPipe));

wait(0); // 等待子进程1 结束

wait(0); // 等待子进程2 结束

int s=read(fd[0],InPipe,sizeof(InPipe)); // 从管道中读出数据

InPipe[s]=0;

printf("%s\n",InPipe);

exit(0); // 父进程结束

}

}

}