实验二、进程通信实验

2.1 实验目的

通过 Linux 系统中管道通信机制，加深对于进程通信概念的理解，观察和体验并发进程间的通信和协作的效果 ，练习利用无名管道进行进程通信的编程和调试技术。

2.2 实验说明

管道 pipe 是进程间通信最基本的一种机制,两个进程可以通过管道一个在管道一端向管道发送其输出,给另一进程可以在管道的另一端从管道得到其输入.管道以半双工方式工作,即它的数据流是单方向的.因此使用一个管道一般的规则是读管道数据的进程关闭管道写入端,而写管道进程关闭其读出端.

⑴ **pipe** 系统调用的语法为**:**

#include <unistd.h>
int pipe(int pipe_id[2]);
如果 pipe 执行成功返回 0, pipe_id[0]中和 pipe_id[1]将放入管道两端的描述符.
出错返回-1.

2.3 示例实验

以下示例实验程序要实现并发的父子进程合作将整数 X 的值从 1 加到 10 的功能。它们通过管道相互将计算结果发给对方。

在新建文件夹中建立以下名为 **ppipe.c** 的 **C** 语言程序

```
/*
   * Filename                : ppipe.c
   * copyright               : (C) 2006 by zhanghonglie
   * Function                : 利用管道实现在父子进程间传递整数
  */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
   int pid;                //进程号
   int pipe1[2];     //存放第一个无名管道标号
   int pipe2[2];    //存放第二个无名管道标号
   int x;              //  存放要传递的整数
      //使用 pipe()系统调用建立两个无名管道。建立不成功程序退出，执行终止
    if(pipe(pipe1) < 0){
        perror("pipe not create");
      exit(EXIT_FAILURE);
```

```
  }
  if(pipe(pipe2) < 0){
     perror("pipe not create");
  exit(EXIT_FAILURE);
  }
  //使用 fork()系统调用建立子进程,建立不成功程序退出，执行终止
if((pid=fork()) <0){
       perror("process not create");
   exit(EXIT_FAILURE);
  }
//子进程号等于 0 表示子进程在执行,
else if(pid == 0){
  //子进程负责从管道 1 的 0 端读,管道 2 的 1 端写，
  //所以关掉管道 1 的 1 端和管道 2 的 0 端。
     close(pipe1[1]);
     close(pipe2[0]);
       //每次循环从管道 1 的 0 端读一个整数放入变量 X 中,
       //并对 X 加 1 后写入管道 2 的 1 端，直到 X 大于 10
       do{
             read(pipe1[0],&x,sizeof(int));
             printf("child %d read: %d\n",getpid(),x++);
             write(pipe2[1],&x,sizeof(int));
       }while( x<=9 );
     //读写完成后,关闭管道
       close(pipe1[0]);
       close(pipe2[1]);
       //子进程执行结束
         exit(EXIT_SUCCESS);
    }
  //子进程号大于 0 表示父进程在执行,
  else{
  //父进程负责从管道 2 的 0 端读,管道 1 的 1 端写，
  //所以关掉管道 1 的 0 端和管道 2 的 1 端。
         close(pipe1[0]);
         close(pipe2[1]);
         x=1;
       //每次循环向管道 1 的 1 端写入变量 X 的值,并从
       //管道 2 的 0 端读一整数写入 X 再对 X 加 1，直到 X 大于 10
         do{
               write(pipe1[1],&x,sizeof(int));
               read(pipe2[0],&x,sizeof(int));
               printf("parent %d read: %d\n",getpid(),x++);
         }while(x<=9);
  //读写完成后,关闭管道
  close(pipe1[1]);
  close(pipe2[0]);
    }
```

```
    //父进程执行结束
     return EXIT_SUCCESS;
}
```

2) 在当前目录中建立以下**Makefile**文件：

```
srcs =    ppipe.c
objs =    ppipe.o
opts =    -g -c
all:        ppipe
ppipe:    $(objs)
          gcc $(objs)    -o ppipe
ppipe.o:  $(srcs)
          gcc    $(opts)    $(srcs)
clean:
          rm ppipe *.o
```

3) 使用**make**命令编译连接生成可执行文件**ppipe:**

$ gmake

```
gcc    -g -c    ppipe.c
 gcc ppipe.o    -o ppipe
```

4) 编译成功后执行**ppipe:**命令**:**

$ ./ppipe

```
child 8697 read: 1
parent 8696 read: 2
child 8697 read: 3
parent 8696 read: 4
child 8697 read: 5
parent 8696 read: 6
child 8697 read: 7
parent 8696 read: 8
child 8697 read: 9
parent 8696 read: 10
```

可以看到以上程序的执行中父子进程合作将整数 X 的值从 1 加到了 10。

2.4 独立实验

设有二元函数 f(x,y) = f(x) + f(y)

其中：

f(x) = f(x-1) * x	(x >1)
f(x)=1	(x=1)
f(y) = f(y-1) + f(y-2)	(y> 2)
f(y)=1	(y=1,2)

请编程建立 3 个并发协作进程，它们分别完成 f(x,y)、f(x)、f(y)