Lab03: Demo Code

管道(Pipe)

管道介绍

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数据（写入管道），而另一个进程可以在管道的另一端从管道读取数据。

管道以半双工的方式工作，即它的数据流是单向的。因此使用管道时的规则一般是读管道数据的进程关闭管道的写入端，而写管道进程关闭其读端口。

管道pipe系统语法说明

pipe系统调用的语法为

```
#include <unistd.h>

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

如果pipe系统调用执行成功，返回0，pipe_id[0]和pipe_id[1]中将放入管道两端的描述符。 出错返回-1。

示例代码

这是一段由并发的父子进程合作将整数X的值从1加到10的示意程序。其中的父子进程开展协作时，两者之间通过管道进行通信。

（1）新建目录oslab3，进入oslab3目录，新建名为ppipe.c的C文件：

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

int main(int argc, char *argv[]) {
    int pid;
    int pipe1[2];
    int pipe2[2];

    int x;

    if (pipe(pipe1) < 0) {
        perror("failed to create pipe1");
        exit(EXIT_FAILURE);
    }
    if (pipe(pipe2) < 0) {
        perror("failed to create pipe2");
        exit(EXIT_FAILURE);
```

```
        }

        pid = fork();
        if (pid < 0) {
            perror("failed to create new process");
            exit(EXIT_FAILURE);
        } else if (pid == 0) {
            // 子进程=>父进程：子进程通过pipe2[1]进行写
            // 子进程<=父进程：子进程通过pipe1[0]读
            // 因此，在子进程中将pipe1[1]和pipe2[0]关闭
            close(pipe1[1]);
            close(pipe2[0]);

            do {
                read(pipe1[0], &x, sizeof(int));
                printf("child %d read: %d\n", getpid(), x++);
                write(pipe2[1], &x, sizeof(int));
            } while (x <= 9);

            close(pipe1[0]);
            close(pipe2[1]);
        } else {
            // 父进程<=子进程:父进程从pipe2[0]读取子进程传过来的数
            // 父进程=>子进程:父进程将更新的值通过pipe1[1]写入，传给子进程
            // 因此，父进程会先关闭pipe1[0]和pipe2[1]端口
            close(pipe1[0]);
            close(pipe2[1]);

            x = 1;

            do {
                write(pipe1[1], &x, sizeof(int));
                read(pipe2[0], &x, sizeof(int));
                printf("parent %d read: %d\n", getpid(), x++);
            } while (x <= 9);

            close(pipe1[1]);
            close(pipe2[0]);
        }

        return EXIT_SUCCESS;
    }
```

(2) 在oslab3中新建Makefile文件，内容如下:

```
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) 在oslab3目录下，执行make命令

```
$ make
```

(4) 编译成功后，执行可执行程序ppipe

```
$ ./ppipe
```

运行结果会显示，父子进程合作将整数x的值从1加到了10。