

实验 30 管道通信

1、匿名管道通信

源程序：

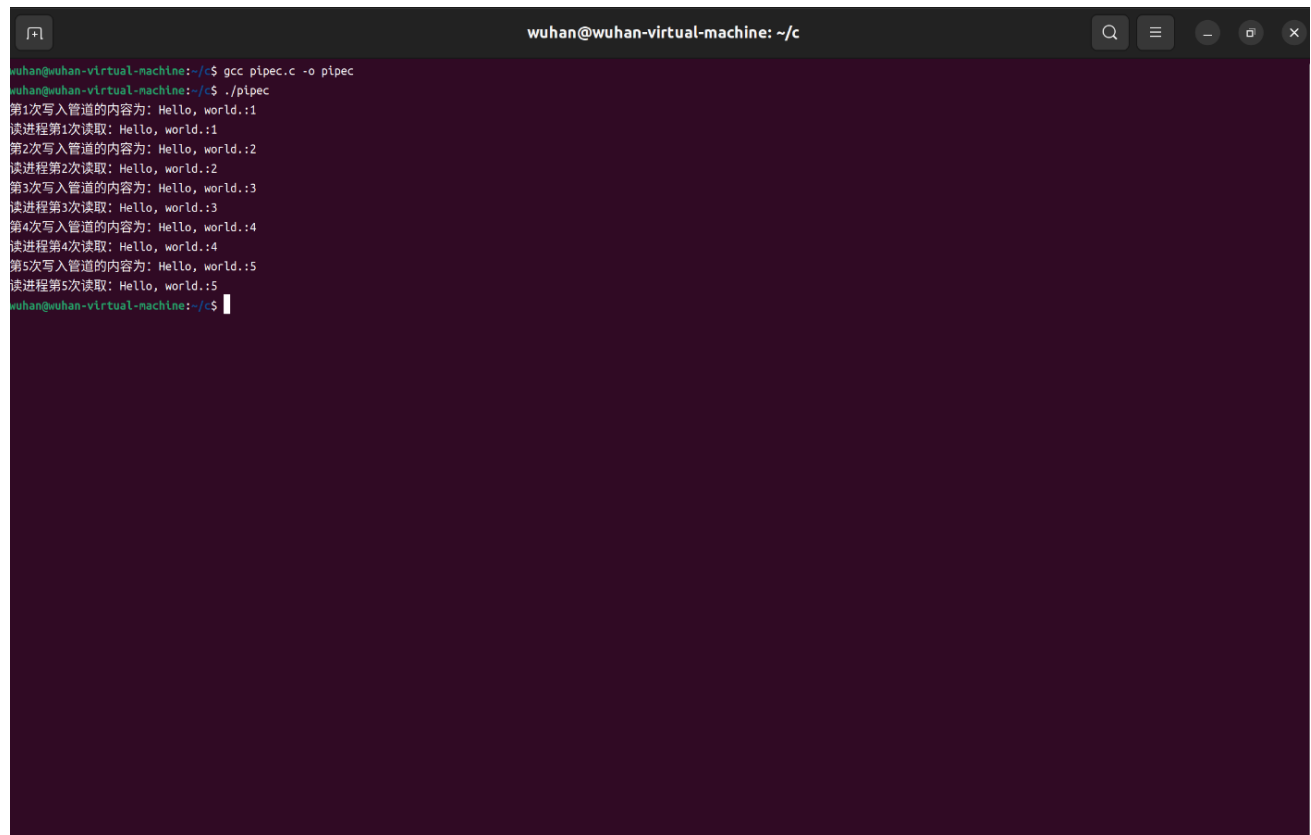
```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
int wc = 1, rc = 1;
void writer(const char *message, int count, FILE *stream)
{
    for (; count > 0; --count)
    {
        fprintf(stream, "%s:%d\n", message, wc);
        sprintf(wstr, "%s:%d\n", message, wc);
        fflush(stream);
        printf("第%d 次写入管道的内容为: %s", wc, wstr);
        wc++;
        sleep(1);
    }
}
void reader(FILE *stream)
{ // 读管道 stream, 次数为 rc
    char buffer[1024];
    while (!feof(stream) && !ferror(stream) && fgets(buffer, sizeof(buffer),
stream) != NULL)
    { // 从管道 stream 读取消息存入 buffer 缓冲区
        printf("读进程第%d 次读取: ", rc);
        rc++;
        fputs(buffer, stdout); // 输出所读管道信息
    }
}
int main()
{
    int fds[2];
    pid_t pid;
    pipe(fds); // 创建匿名管道
    pid = fork(); // 创建子进程
    if (pid == (pid_t)0)
    { // 子进程代码
        FILE *stream;
        close(fds[1]); // 关闭管道写端
        stream = fdopen(fds[0], "r"); // 子进程以只读方式打开管道
        reader(stream); // 子进程读取管道
        close(fds[0]); // 子进程关闭读端
    }
    else
    { // 父进程代码
        FILE *stream;
        close(fds[0]); // 关闭管道读端
        stream = fdopen(fds[1], "w"); // 父进程以只写方式打开管道
        writer("Hello, world.", 5, stream); // 父进程向管道写入 5 次消息“Hello,
world.”
    }
}
```

```
        close(fds[1]);                // 子进程关闭写端
    }
    return 0;
}
```

编译链接命令: `gcc pipec.c -o pipec`

运行命令: `./pipec`

交互与结果:



```
wuhan@wuhan-virtual-machine: ~/c
wuhan@wuhan-virtual-machine:~/c$ gcc pipec.c -o pipec
wuhan@wuhan-virtual-machine:~/c$ ./pipec
第1次写入管道的内容为: Hello, world.:1
读进程第1次读取: Hello, world.:1
第2次写入管道的内容为: Hello, world.:2
读进程第2次读取: Hello, world.:2
第3次写入管道的内容为: Hello, world.:3
读进程第3次读取: Hello, world.:3
第4次写入管道的内容为: Hello, world.:4
读进程第4次读取: Hello, world.:4
第5次写入管道的内容为: Hello, world.:5
读进程第5次读取: Hello, world.:5
wuhan@wuhan-virtual-machine:~/c$
```

2、有名管道通信

源程序：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#define FIFO "myfifo" // 有名管道名字
#define BUFF_SIZE 1024
int main()
{
    char buff[BUFF_SIZE]; // 欲读取管道的数据缓冲区
    int real_read;         // 读取管道的字节数
    int fd;                // 管道描述符
    int rc = 1;            // 管道读次数
    if (access(FIFO, F_OK) == -1)
    { // 测试有名管道 FIFO 是否存在，若不存在，则用 mkfifo 创建该管道
        if ((mkfifo(FIFO, 0666) < 0) && (errno != EEXIST))
        { // 创建管道"myfifo", 允许读写
            printf("Can NOT create fifo file!\n");
            exit(1);
        }
    }
    if ((fd = open(FIFO, O_RDONLY)) == -1)
    { // 以只读方式打开 FIFO, 返回文件描述符 fd
        printf("Open fifo error!\n");
        exit(1);
    }
    while (1)
    { // 循环读管道，若读空，则结束循环
        memset(buff, 0, BUFF_SIZE);
        if ((real_read = read(fd, buff, BUFF_SIZE)) > 0)
            printf("第%d 次读取管道: '%s'.\n", rc++, buff);
        else
            break;
    }
    close(fd);
    exit(0);
}
```

编译链接命令：

gcc fifo_write.c -o fifo_write

gcc fifo_read.c -o fifo_read

运行命令：

./fifo_write

./fifo_read

交互与结果:

The image shows a Linux desktop environment with two terminal windows. The left window is titled 'wuhan@wuhan-virtual-machine: ~/c' and shows the execution of the `./fifo_write` program. The right window is also titled 'wuhan@wuhan-virtual-machine: ~/c' and shows the execution of the `./fifo_read` program. Both windows display the same sequence of inputs and outputs, demonstrating the FIFO (First In, First Out) behavior of the pipe.

```
wuhan@wuhan-virtual-machine: ~/c
wuhan@wuhan-virtual-machine:~/c$ cd c
wuhan@wuhan-virtual-machine:~/c$ ./fifo_write
请输入要写入管道的内容: wuhan 202101000720
第1次写入管道: 'wuhan 202101000720'.
请输入要写入管道的内容: wuhan handsome
第2次写入管道: 'wuhan handsome'.
请输入要写入管道的内容: OS exp
第3次写入管道: 'OS exp'.
请输入要写入管道的内容: hihihi
第4次写入管道: 'hihihi'.
请输入要写入管道的内容:

wuhan@wuhan-virtual-machine: ~/c
wuhan@wuhan-virtual-machine:~/c$ cd c
wuhan@wuhan-virtual-machine:~/c$ ./fifo_read
第1次读取管道: 'wuhan 202101000720'.
第2次读取管道: 'wuhan handsome'.
第3次读取管道: 'OS exp'.
第4次读取管道: 'hihihi'.
```