

山东大学 计算机科学与技术 学院

操作系统 课程实验报告

学号：202100130022	姓名：郭家宁	班级：21 数据
实验题目：实验 2：线程和管道通信		
实验学时：2	实验日期：2023-4-17	
实验目的：通过 Linux 系统中线程和管道通信机制的实验，熟悉 pthread 线程库的使用，加深对于线程控制和管道通信概念的理解，观察和体验并发线程间的通信和协作的效果，练习基于 pthread 线程库、利用无名管道进行线程通信的编程和调试技术。		

实验结果：

2.2 利用管道实现在线程间传递整数

利用两个管道在两个线程之间传递参数

```
jianing@jianing-virtual-machine:~/桌面/os_ex/exe2/exe2_example/tpipe_test$ ./tpipe
140430977005120 140430968612416
thread2 read: 1
thread1 read: 2
thread2 read: 3
thread1 read: 4
thread2 read: 5
thread1 read: 6
thread2 read: 7
thread1 read: 8
thread2 read: 9
thread1 read: 10
jianing@jianing-virtual-machine:~/桌面/os_ex/exe2/exe2_example/tpipe_test$
```

2.3 利用管道实现在父子进程间传递整数

子进程 8590 和父进程 8589 交替执行，利用两个无名管道在子父进程之间传递参数。

```
jianing@jianing-virtual-machine:~/桌面/os_ex/exe2/exe2_example/ppipe_test$ ./ppipe
child 8590 read: 1
parent 8589 read: 2
child 8590 read: 3
parent 8589 read: 4
child 8590 read: 5
parent 8589 read: 6
child 8590 read: 7
parent 8589 read: 8
child 8590 read: 9
parent 8589 read: 10
jianing@jianing-virtual-machine:~/桌面/os_ex/exe2/exe2_example/ppipe_test$
```

2.4 独立实验：

设有二元函数 $f(x, y) = f(x) + f(y)$

其中：

- $f(x)=1 \quad (x=1)$

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

请基于无名管道，利用 pthread 线程库编程建立 3 个并发协作线程，它们分别完成 $f(x, y)$ 、 $f(x)$ 、 $f(y)$ 系统调用

实验思路：

(1)

使用 fork() 系统调用，建立两个子进程，然后分别让子进程执行 $f(x)$ ， $f(y)$ ，然后建立两个通道，让子进程给父进程传递参数 $f(x)$ ， $f(y)$ ，，然后让父进程计算 $f(x, y)$ 。

子进程 1 在管道 1 的 1 端写数据，所以要关闭管道 1 的 0 端，在 pipe1[1] 写入数据。

```
if (pid1 == 0) // 子进程执行f(x)
{
    close(pipe1[0]); // 子进程从管道1的1端进行写数据

    int result_x = task1(x);

    printf("f(x) = %d\n", result_x);
    printf("child %d write : %d\n", getpid(), task1(x));

    write(pipe1[1], &result_x, sizeof(int));

    // 写完成后,关闭管道
    close(pipe1[1]);
}
```

子进程 2 在管道 2 的 1 端写数据,所以要关闭管道 2 的 0 端,在 pipe2[1] 写入数据.

```
if (pid2 == 0) // 子进程2执行f(y)
{
    close(pipe2[0]); // 子进程从管道1的1端进行写数据

    int result_y = task2(y);

    printf("f(y) = %d\n", result_y);
    printf("child %d write : %d\n", getpid(), result_y);

    write(pipe2[1], &result_y, sizeof(int));

    // 写完成后,关闭管道
    close(pipe2[1]);
}
```

父 进 程 从 pip1[0] 和 pip2[0] 读 取 数 据 。

```
else
{
    waitpid(pid2, &status2, 0); // 等待子进程2结束

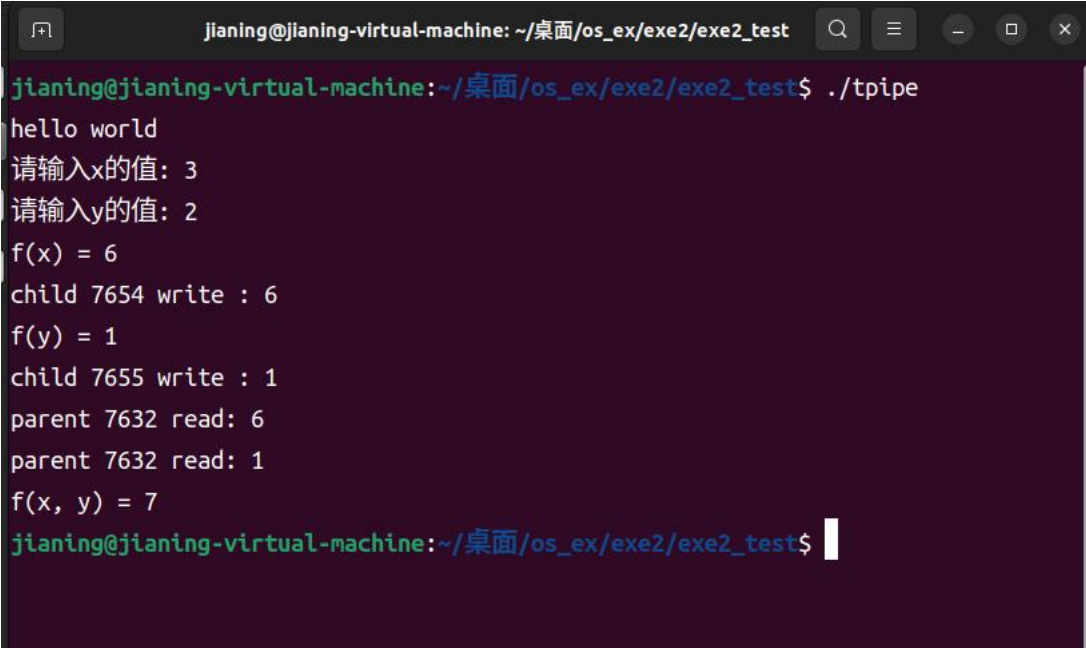
    // 父进程分别冲两个管道的0端读取子进程读取出来的信息

    // 子进程1
    close(pipe1[1]);
    int tempx;
    read(pipe1[0], &tempx, sizeof(int));
    printf("parent %d read: %d\n", getpid(), tempx);
    close(pipe1[0]);

    // 子进程2
    close(pipe2[1]);
    int tempy;
    read(pipe2[0], &tempy, sizeof(int));
    printf("parent %d read: %d\n", getpid(), tempy);
    close(pipe2[0]);

    int result = task3(tempx, tempy);
    printf("f(x, y) = %d\n", result);
}
```

最后执行结果如下，发现子进程 7654 执行 $f(x)$ ，写入 6，子进程 7655 执行 $f(y)$ ，写入 1，最后父进程读取 6 和 1，然后执行 $f(x, y)$ ，得出结果。



```
jianing@jianing-virtual-machine: ~/桌面/os_ex/exe2/exe2_test
jianing@jianing-virtual-machine:~/桌面/os_ex/exe2/exe2_test$ ./tpipe
hello world
请输入x的值: 3
请输入y的值: 2
f(x) = 6
child 7654 write : 6
f(y) = 1
child 7655 write : 1
parent 7632 read: 6
parent 7632 read: 1
f(x, y) = 7
jianing@jianing-virtual-machine:~/桌面/os_ex/exe2/exe2_test$
```

2.5 实验要求

- 根据示例实验程序和独立实验程序观察和记录的调试和运行的信息，说明它们反映出操作系统教材中讲解的进/线程协作和进/线程通信概念的哪些特征和功能？

进/线程协作：进/线程之间可以通过协作来完成某项任务。一个进程需要等待另一个进程的数据处理结果才能继续执行，这就需要它们之间进行协作。

进/线程通信：进/线程之间可以通过通信来共享信息和数据。不同的进程可以通过不同形式的通信方式进行交流。

- 在真实的操作系统中它是怎样实现和反映出教材中进/线程通信概念的。你对于进/线程协作和进/线程通信的概念和实现有哪些新的理解和认识？

在真实的操作系统中，进/线程通过管道实现通信，管道是一种双向通信机制，它允许两个进程之间进行通信，并且数据流只能从一个方向流动。在管道中，数据被传递到管道的读取端口，然后再通过管道连接的写入端口传递到接收进程。

进/线程协作和进/线程通信能够有效地提高系统效率和资源利用率。同时，在实际的开发中，需要根据具体情况选择合适的协作和通信机制，可以很好的提高系统的性能。

- 管道机制的机理是什么？怎样利用管道完成进/线程间的协作和通信？

管道机制的机理是，管道是一个缓存区，它分为读取端口和写入端口。数据被写入管道，然后通过管道传输到读取端口，并从读取端口被接收。管道可以实现两个进程之间的通信，它是一种可靠的双向通信机制，

问题及收获：

加深了对线程控制和管道通信概念的理解，以及对并发线程间的通信和协作的认识