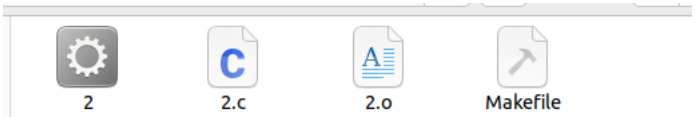
山东大学 计算机科学与技术 学院
操作系统 课程实验报告

学号：202200130223	姓名：李洋永	班级：4 班
实验题目：实验 2 线程和管道通信		
实验学时：2	实验日期：2024/10/16	
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。		
实验环境：VirtualBox、Ubuntu		
源程序清单： 		
编译及运行结果： q@q-virtual-machine:~/Desktop/lab2/ex2$ make make: 对“all”无需做任何事。 q@q-virtual-machine:~/Desktop/lab2/ex2$ ./2 请输入xy: 2 3 f(x)=2 f(y)=2 f(x+y)=4 q@q-virtual-machine:~/Desktop/lab2/ex2$ ./2 请输入xy: 3 4 f(x)=6 f(y)=3 f(x+y)=9 q@q-virtual-machine:~/Desktop/lab2/ex2$ ./2 请输入xy: 4 5 f(x)=24 f(y)=5 f(x+y)=29 q@q-virtual-machine:~/Desktop/lab2/ex2$ ./2 请输入xy: 5 6 f(x)=120 f(y)=8 f(x+y)=128 q@q-virtual-machine:~/Desktop/lab2/ex2$		

```c
void task1(int *x)
{
    int fx = 1;
    for (int i = 1; i <= *x; i++) fx *= i;
    printf("f(x)=%d\n", fx);
    write(pipe1[1], &fx, sizeof(int));
    close(pipe1[1]);

}

//线程 2 执行函数
void task2(int * y)
{
    double q1 = (1 + sqrt(5)) / 2, q2 = (1 - sqrt(5)) / 2;
    int fy = (pow(q1, *y) - pow(q2, *y)) / sqrt(5);
    printf("f(y)=%d\n", fy);
    write(pipe2[1], &fy, sizeof(int));
    close(pipe2[1]);
}
void task3(int *x)
{
    int fx, fy;
    read(pipe1[0], &fx, sizeof(int));
    read(pipe2[0], &fy, sizeof(int));
    printf("f(x+y)=%d\n", fx + fy);
    close(pipe1[0]);
    close(pipe2[0]);
}
```

在 main 函数中建立线程，task1 算出 f（x）结果，写入 pipe1，task2 算出 f（y）结果，写入 pipe2，

Task3 从管道读取数据输出 f（x+y）

问题及收获:

```
    pthread_join(thrd2,NULL);
    //挂起当前线程，等待线程 thrd1 结束，并回收其资源
    pthread_join(thrd1,NULL);
    //思考与测试：如果去掉上述两个 pthread_join 的函数调用，会出现什么现象
    exit(EXIT_SUCCESS);
}
```

pthread_join 让主线程等待子线程的终止，如果去掉主线程直接结束，不执行子线程 thrd1 和 thrd2

多个进程或线程可以为了完成共同的任务而相互配合，线程间可以通过管道传递信息。进程/线程协作和通信是现代操作系统中实现并发和并行处理的核心机制。协作机制通过同步来保证多个进程/线程可以正确地共享资源和共同完成任务，而通信机制则通过不同的数据交换方式实现进程/线程间的信息传递和任务协调。

管道通常是单向的，允许一个进程在管道写入数据，另一进程读取数据。

以实验样例 tpipe 为例，建立两个管道，线程 1 写入管道 1，线程 2 读出数据并加 1 后写入管道 2，由于管道读写是阻塞的，线程 1 在管道 2 有数据时才读出数据继续运行，线程 1、2 数据是同步的，这样可以实现协作和通信。

```
void task1(int *num)
{
    int 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("thread%d read: %d\n",*num,x++);
    }while(x<=9);

    //读写完成后,关闭管道
    close(pipe1[1]);
    close(pipe2[0]);
}
//线程 2 执行函数，它首先从管道 1 读，然后向管道 2 写
void task2(int * num)
{
    int x;
    //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("thread2 read: %d\n",x++);
        write(pipe2[1],&x,sizeof(int));
    }while( x<=9 );
```