实验3：在xv6上使用管道实现“质数筛选”, 输出2~35之间的而所有质数。请将代码写在user/primes.c文件中。

1、程序设计

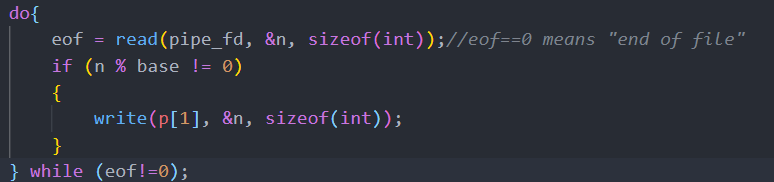
思路：

该程序本质上是“合作进程”的设计模式。也就是每个父进程开启一个或多个子进程来进行协作处理，并利用管道进行参数传入和结果返回。

这也是一个递归的过程。每次父进程fork出一个子进程，就类似于在递归算法里面产生了一个新的函数栈帧。所以程序的关键在于递归出口和递归链的设计。

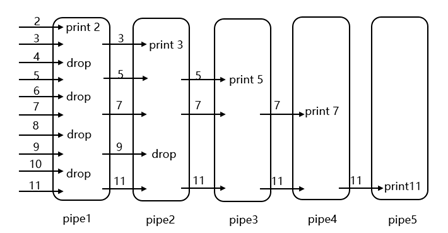
递归出口：上一次递归的子进程执行read(pipe\_fd, &base, sizeof(int)) == 0如果从pipe中读不到数，说明递归已经结束了。

递归链：



对父进程输入的数进行处理并传递下去。其中base是做质数检验的参数。

示意图：



完整代码：

#include "kernel/types.h"

#include "user/user.h"

void prime(int *pipe\_fd*)

{

*// 从pipe读到的首个数。*

    int base;

*// 如果从pipe中读不到数，说明递归已经结束了*

    if (read(pipe\_fd, &base, sizeof(int)) == 0)

    {

        exit(0);

    }

    printf("prime %d\n", base);

*// 否则进行递归*

    int p[2];

    pipe(p);

    if (fork() == 0)*//*

    {

        close(p[1]);

        prime(p[0]);

    }

    else

    {

        close(p[0]);

        int n;

        int eof;

        do

        {

            eof = read(pipe\_fd, &n, sizeof(int));

            if (n % base != 0)

            {

                write(p[1], &n, sizeof(int));

            }

        } while (eof);

        close(p[1]);

    }

    wait(0);

    exit(0);

}

int main(int *argc*, char const \**argv*[])

{

    int origin\_fd[2];

    pipe(origin\_fd);

    if (fork()>0)*//parent; main;*

    {

        close(origin\_fd[0]);

        int i;

        for (i = 2; i < 36; i++)*// 针对主进程，枚举2~35传给子进程筛选质数。*

        {

            write(origin\_fd[1], &i, sizeof(int));

        }

        close(origin\_fd[1]);

    }

    else

    {

        close(origin\_fd[1]);

        prime(origin\_fd[0]);

    }

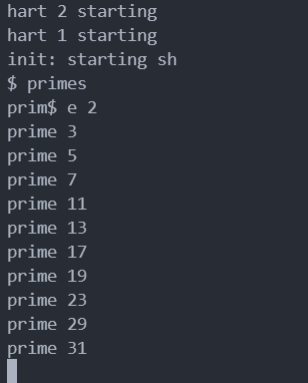
    wait(0);

    exit(0);

}

2、处理Bug

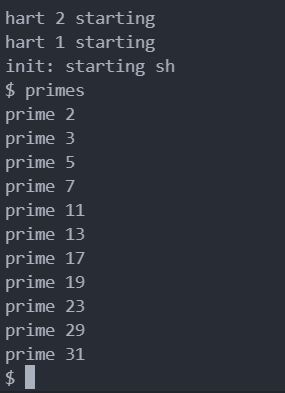
结果出现了“$符号已经打印出来了，但是子进程还在运行状态中。”这肯定是不符合要求的。



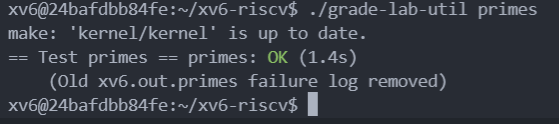
查了一下提示，说要“要确保子进程先退出，父进程再退出。”所以这里使用wait系统调用。

wait函数用在父进程中能保证子进程在父进程结束前被析构回收，保证了进程的执行次序，就不会产生合作进程之间不同步的问题了。

程序运行结果如下：



自动化测试的结果：



通过测试！

本实验没有问题需要回答。