Lab: Xv6 and Unix utilities实验报告

Git链接：[Jeery1/xv6-6.S081: 2019版的xv6实验实现](https://github.com/Jeery1/xv6-6.S081/tree/main)

1. sleep

**实验目的：**

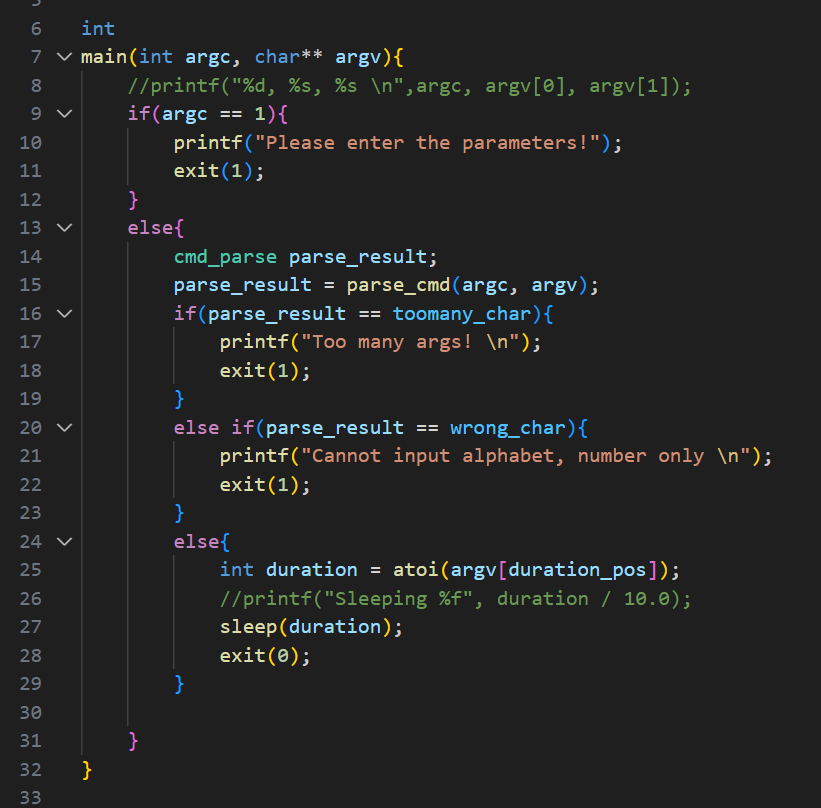
实现基本功能：创建一个用户程序 sleep，能够暂停执行指定的时钟滴答数。正确处理命令行参数（将字符串参数转换为整数）使用xv6提供的 sleep 系统调用

错误处理：当用户忘记提供参数时，打印错误信息确保程序在各种情况下都能正确退出

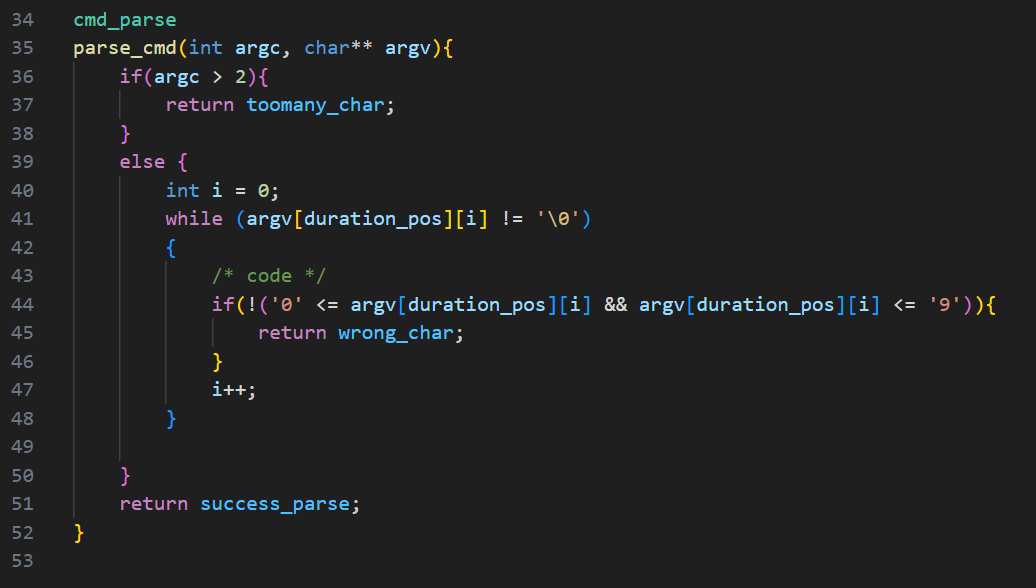
系统集成：将 sleep 程序正确添加到xv6的Makefile中。确保编译后能在xv6 shell中运行

**实验步骤：**

在user文件夹中新建一个sleep.c文件，编写实现sleep指令相关的代码。然后修改makefile文件，将sleep.c加入。然后执行make clean和make qemu命令，进入qemu环境，验证sleep命令的准确性。

**代码分析：**

检查参数数量，如果没有参数就输出错误信息。然后将参数解析逻辑分离到单独函数。检查参数是否过多或是否参数输入错误（不是数字），输出相应错误信息。如果没有错误就进入正常执行sleep逻辑



参数解析函数，首先检查参数数量是否过多（大于两个）。然后严格检查每个字符是否为数字。

**实验现象：**

****

正确输入，等待一小段时间后又出现$



输入字母作为参数



没有输入参数



参数过多

1. pingpong

**实验目的：**

管道通信实践：创建两个单向管道（parent→child和child→parent）实现父子进程间双向字节传输

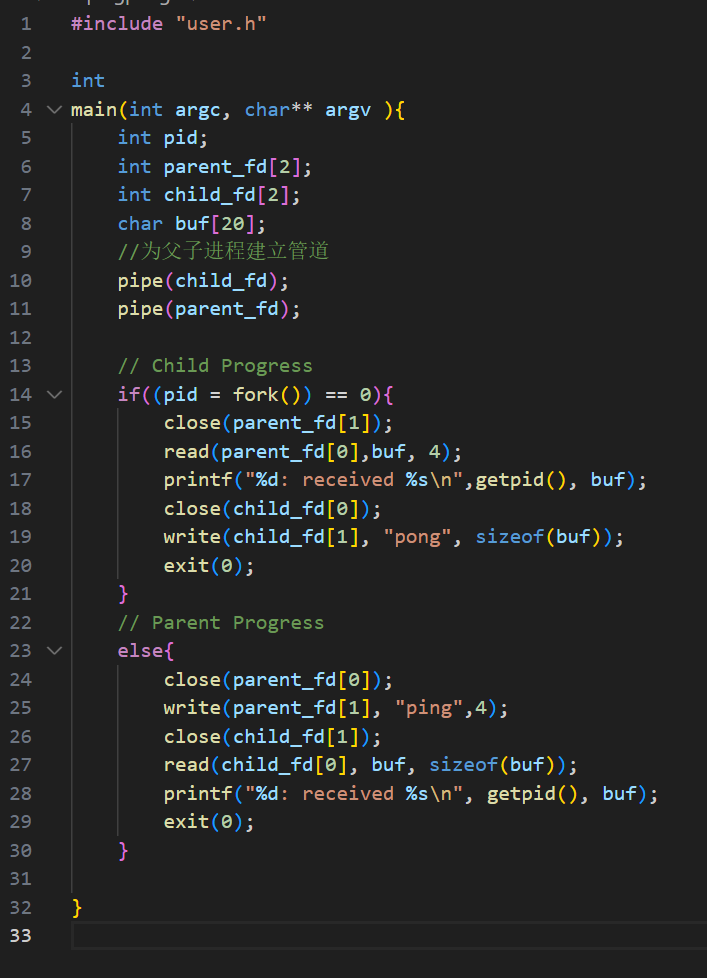
进程协作：父进程先发送"ping"字节，子进程接收后回复"pong"字节按指定格式打印进程ID和接收到的消息

系统调用运用：正确使用pipe()创建管道，通过fork()创建子进程，使用read()/write()进行管道I/O操作，调用getpid()获取进程ID

**实验步骤：**

在user文件夹中新建一个pingpong.c文件，编写实现pingpong指令相关的代码。然后修改makefile文件，将pingpong.c加入。然后执行make clean和make qemu命令，进入qemu环境，验证pingpong命令的准确性。

**代码分析：**



创建了两个独立的管道：

child\_fd：子进程写，父进程读

parent\_fd：父进程写，子进程读

每个管道提供两个文件文件描述符：[0]：读取端 [1]：写入端

子进程逻辑分析：

首先关闭不需要的管道端（避免资源泄漏），遵循"读关闭写，写关闭读"的原则

通信流程：

先读取父进程发送的"ping"消息

打印接收到的消息

向父进程发送"pong"响应

使用getpid()获取并打印当前进程ID

父进程逻辑分析：  
文件描述符管理：

对称地关闭不需要的管道端

保持与子进程相反的文件描述符开关状态

通信流程：

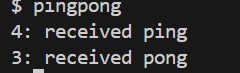
先向子进程发送"ping"消息

等待并读取子进程的"pong"响应

打印接收到的消息

进程同步：通过管道的阻塞读写实现自然的进程同步

**实验现象：**



如果pingpong是进入qemu执行的第一个指令，那么两个进程的ID必定是3和4。这是因为0、1、2号ID都分配给了特殊的进程。0号进程是闲逛进程，1号进程是所有用户空间进程的祖先，2号进程是内核线程管理器。

1. primes

**实验目的：**

并发算法实现：

使用管道和进程构建一个并发的素数筛选管道

每个素数对应一个独立的过滤进程

资源管理：

正确处理文件描述符（及时关闭不需要的管道端）

在xv6有限的资源限制下完成计算（2-35范围内的素数）

进程协作：

主进程输入数字2-35到管道

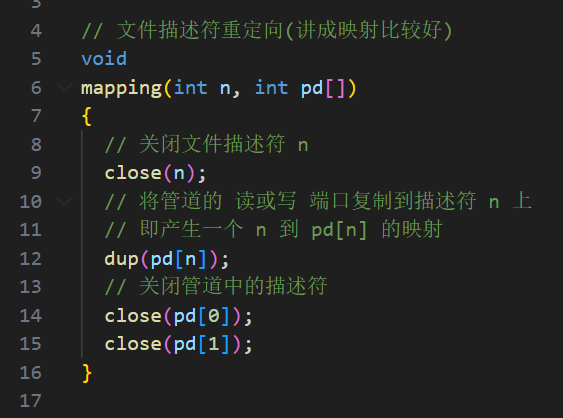
每个素数进程过滤其倍数的数字

实现优雅的管道终止机制

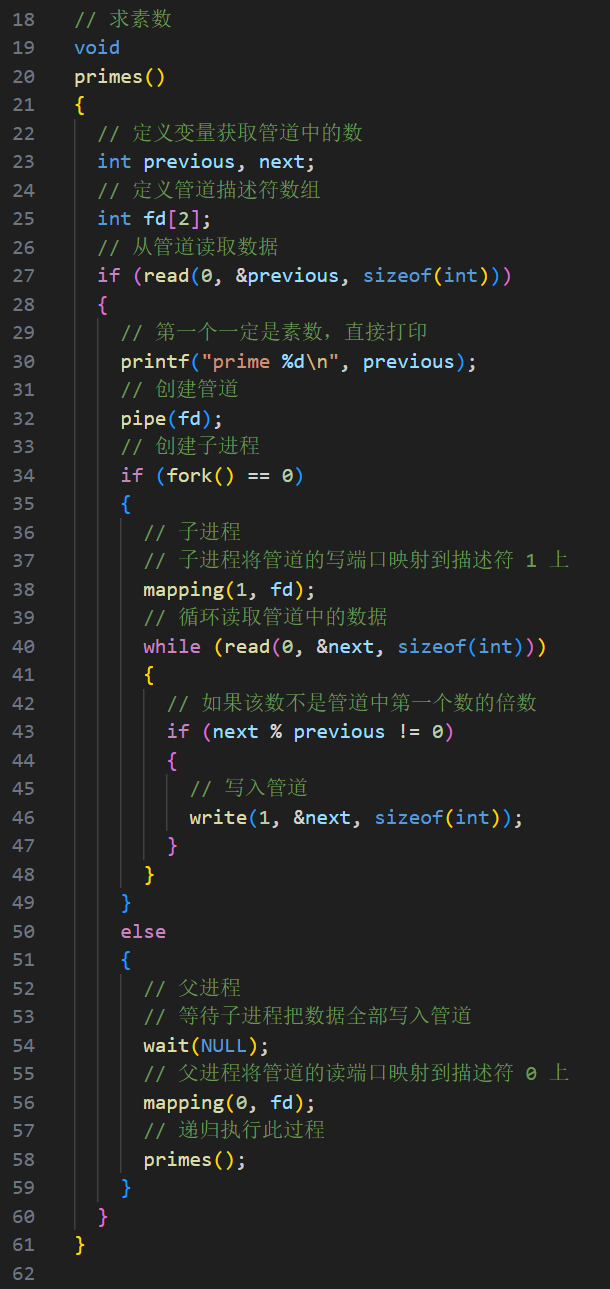
**实验过程：**

在user文件夹中新建一个primes.c文件，编写实现prime指令相关的代码。然后修改makefile文件，将primes.c加入。然后执行make clean和make qemu命令，进入qemu环境，验证primes命令的准确性。

**代码分析：**



Mapping函数，将管道的读/写端重定向到标准输入/输出，清理不需要的文件描述符



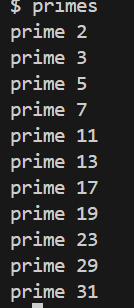
Prime函数，通过递归调用进程筛选素数。

筛选逻辑：读取的第一个数必然是素数（打印它）然后创建新管道和子进程，子进程过滤当前素数的倍数，父进程递归处理剩余数字

进程管理：每个素数对应一个过滤进程，通过递归创建进程管道

通信机制：使用管道传递数字，依赖文件描述符重定向建立通信链

**实验现象：**



输出1到35的所有质数

1. find

**实验目的：**

目录遍历实现：

递归遍历指定目录及其子目录

正确跳过"."和".."目录项

文件名匹配：

实现基础字符串匹配功能

文件系统操作：

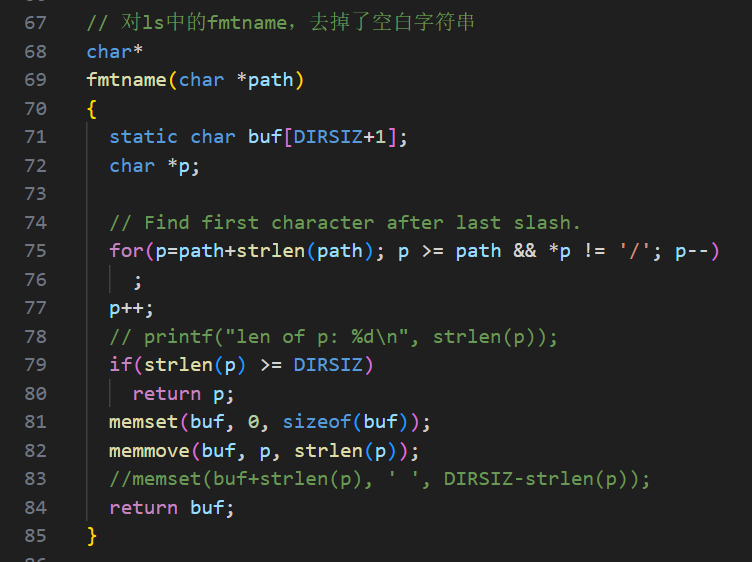
使用xv6的文件系统系统调用

正确处理文件类型（普通文件/目录）

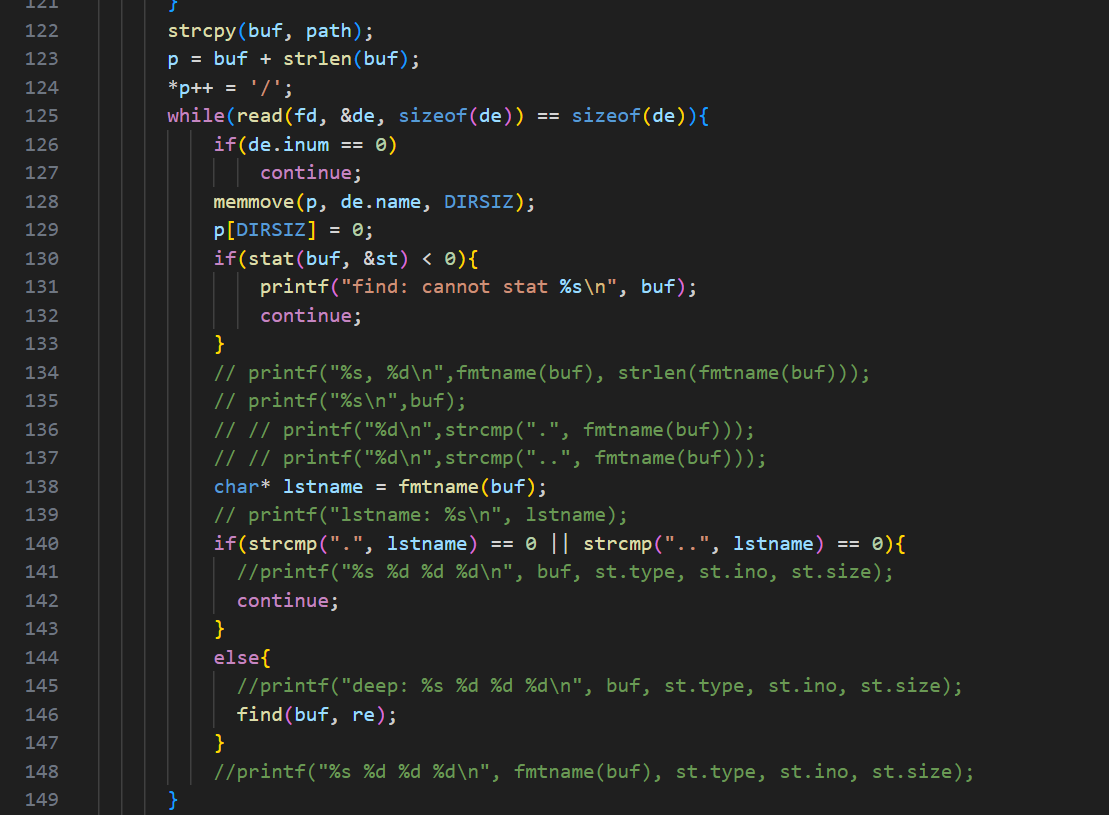
**实验步骤：**

在user文件夹中新建一个find.c文件，编写实现find指令相关的代码。然后修改makefile文件，将find.c加入。然后执行make clean和make qemu命令，进入qemu环境，验证find命令的准确性。

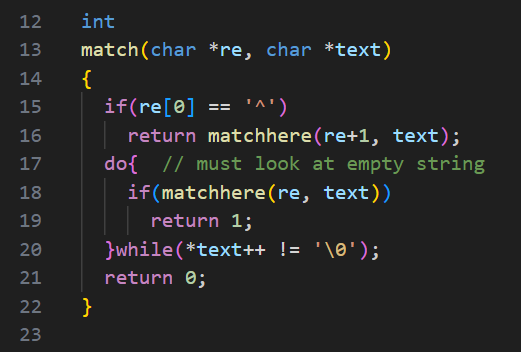
**代码分析：**



Fmtname函数，剥离路径只保留文件名部分，使用静态缓冲区避免内存分配，处理xv6文件系统的DIRSIZ(14字节)限制。

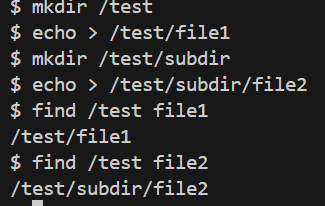


手动构建路径字符串，使用DIRSIZ(14字节)限制处理短文件名，显式跳过"."和".."目录，递归调用实现深度优先搜索。

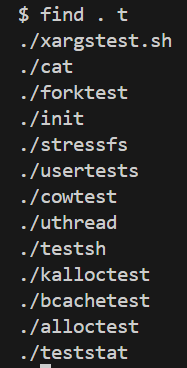


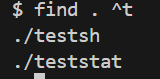
Match函数，实现字符串匹配

**实验现象：**



建立test文件夹，其中包括file1和subdir文件夹。subdir文件夹里面有file2，使用find指令在test目录下查找file1和file2，可以看到能够正常输出文件的路径。





使用find . t可以查找当前目录下所有名字包含t的文件，使用find . ^t可以查找以名字以t开头的文件。

1. xargs

**实验目的：**

输入处理：

动态读取标准输入（一行一行地读）

处理任意长度的行（需合理管理缓冲区）

参数构造：

将输入行拆分为多个参数（如空格分隔）

合并用户输入的参数和从 stdin 读取的参数

命令执行：

对每一行输入生成新进程执行命令

确保参数列表以 NULL 结尾（符合 exec 要求）

错误处理：

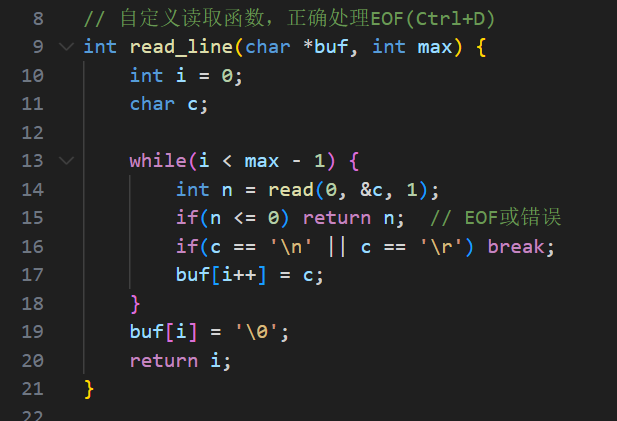
处理 fork/exec 失败的情况

避免内存泄漏或文件描述符泄漏

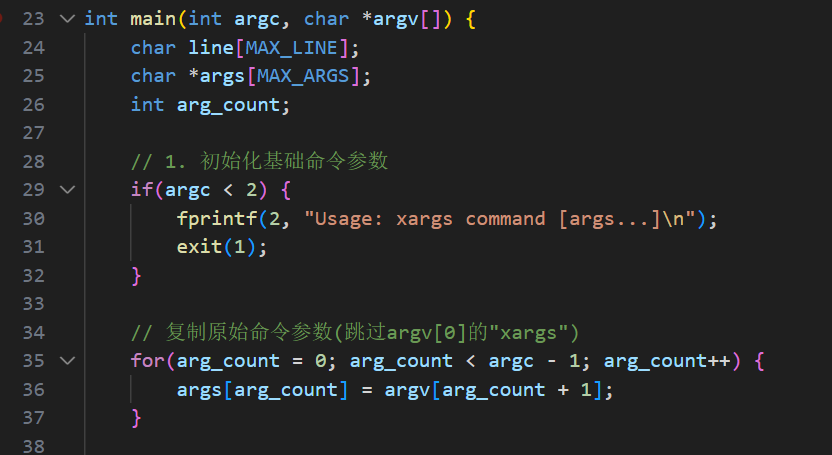
**实验步骤：**

在user文件夹中新建一个xarg.c文件，编写实现xarg指令相关的代码。然后修改makefile文件，将xarg.c加入。然后执行make clean和make qemu命令，进入qemu环境，验证xarg命令的准确性。

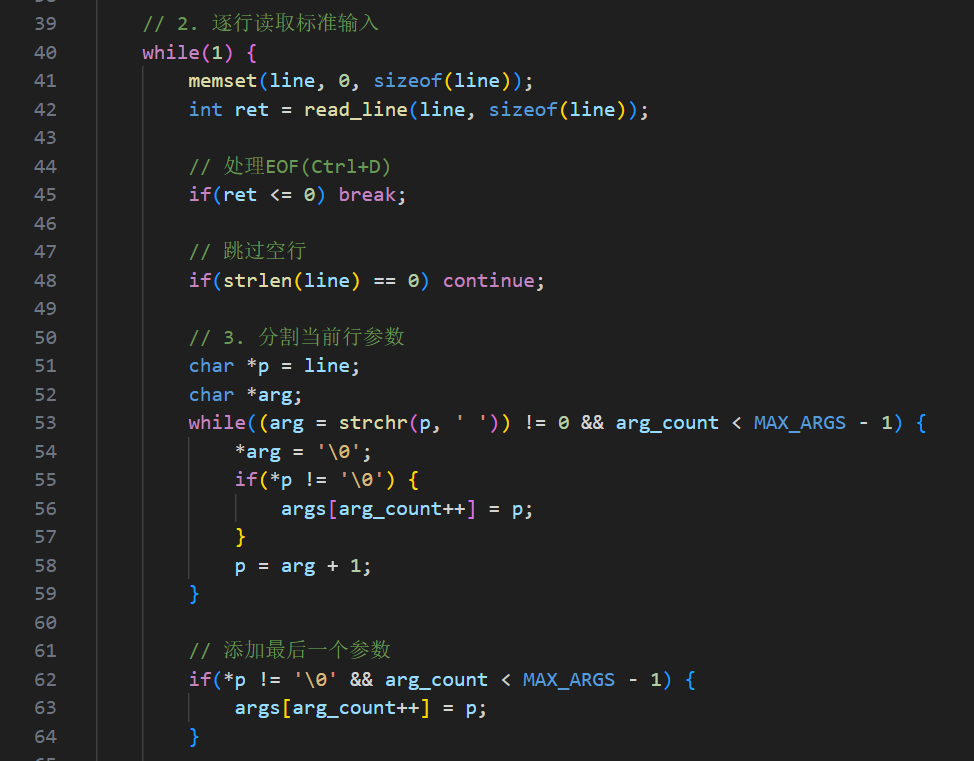
**代码分析：**



Readline函数，逐字节读取确保精确控制，返回值处理：n<=0 表示EOF（Ctrl+D）或错误，自动跳过回车符（\r）兼容不同系统。

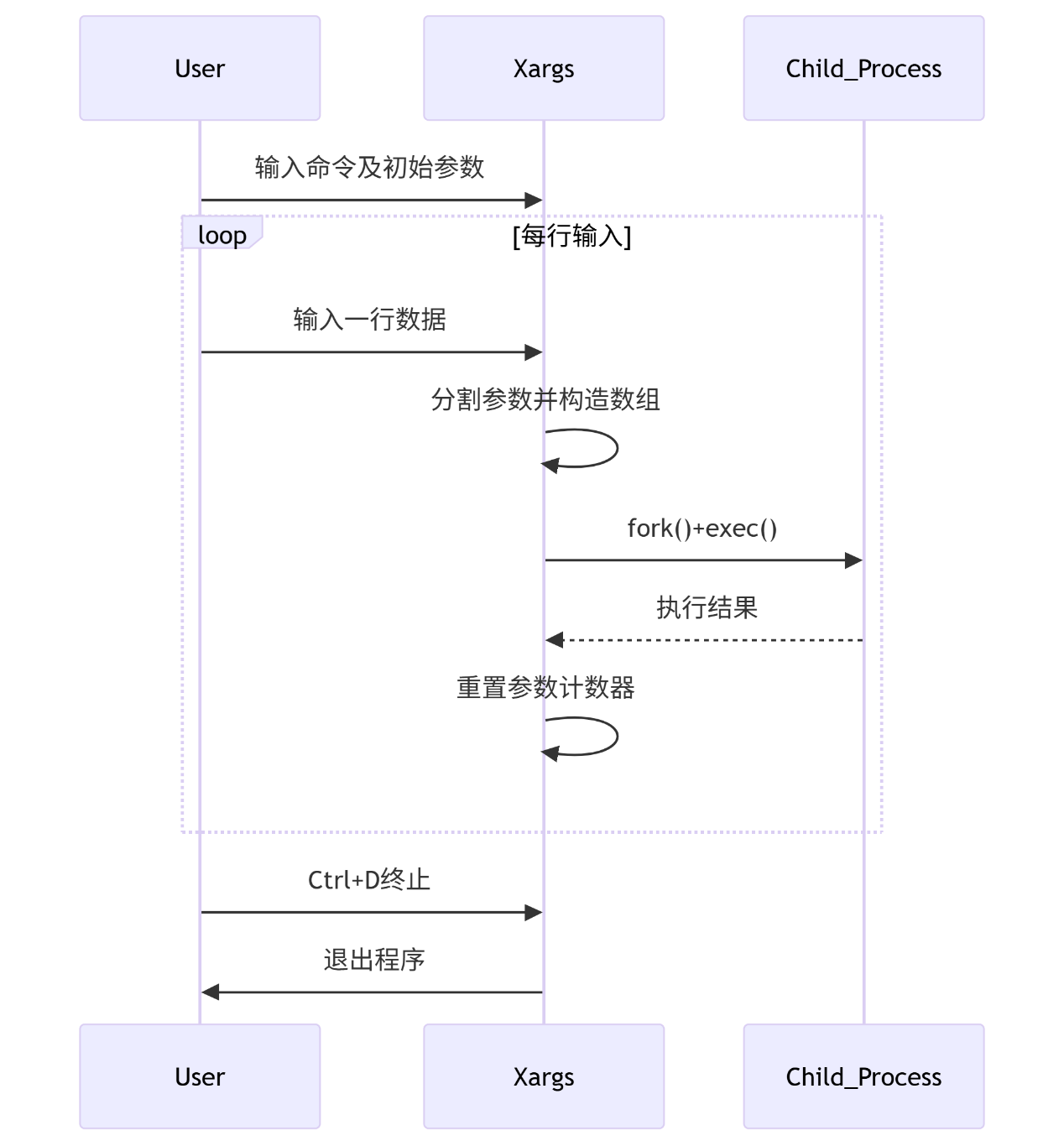


主函数，首先将 xargs 后的命令+参数存入 args 数组

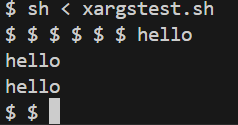


从输入缓冲区循环读取数据，如果读取到截止符（ctrl+D）就停止读取。将读取到的数据加入args数组作为参数。

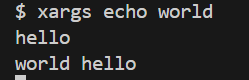
流程图：



**实验现象：**



执行自带的测试程序，通过。



可以看出，在缓冲区输入的hello作为参数加在了echo world命令的后面

**实验中遇到的问题和解决办法**

在实现sleep命令时，最初编写的代码如下：

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

if (argc != 2) {

fprintf(2, "Usage: sleep ticks\n");

exit(1);

}

sleep(atoi(argv[1]));

exit(0);

}

编译时出现错误："undefined reference to 'sleep'"。

原因分析：

Xv6是一个简化的操作系统，其系统调用与标准Unix有所不同。直接使用标准C库的sleep函数会导致链接错误，因为Xv6没有提供这个函数。

解决办法：

使用Xv6提供的系统调用接口。正确的实现应该包含Xv6特定的头文件user/user.h，它提供了对Xv6系统调用的封装。

在实现primes筛法程序时，随着递归深度增加，系统最终耗尽内存。

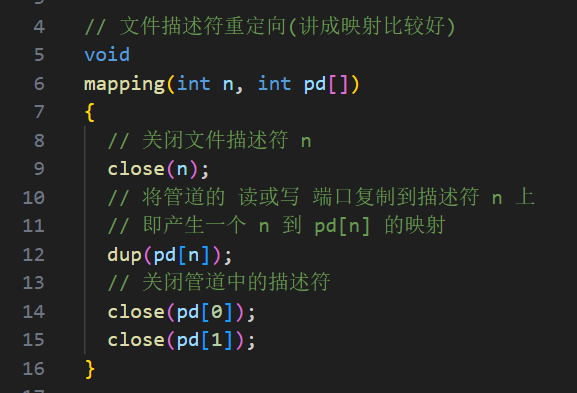
处理小数字时程序运行正常，当数字范围增大时，程序崩溃。使用top命令观察内存使用情况，发现内存持续增长。

原因分析：

每个递归层级都创建了新的管道但没有正确关闭所有文件描述符，导致文件描述符泄漏。

解决办法：

确保在每个进程结束时关闭所有不需要的文件描述符。



**实验心得：**

通过本次实验，我深入理解了Unix实用程序的基本工作原理和Xv6操作系统的特点。遇到的各种问题让我更加清楚了文件描述符管理、进程通信和系统调用等核心概念的重要性。特别是在处理管道通信和递归目录搜索时，需要特别注意资源管理和错误处理，这些经验对后续实验的学习有很大帮助。