****



**MIT xv6实验报告**

**——lab1 : util**

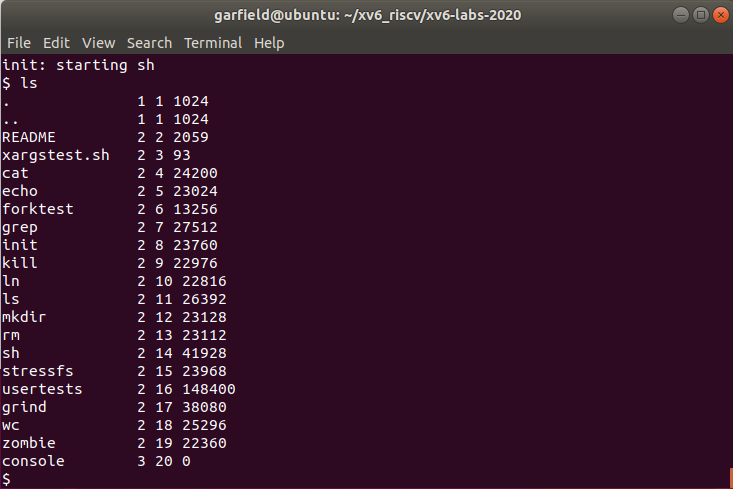
**学生姓名 胡轶然**

**学 号 3019244355**

1. **实验目的**
2. **前期准备**

1. 配置环境，boot xv6

1. 按要求使用git工具clone项目，并进入指定分支，然后执行make qemu命令，结果与指导书相符



**三、** **实验内容及实现步骤**

**任务1 sleep**

**1. 问题描述**

实现sleep命令，当输入sleep n时，休眠一段时间。

**2. 思路与实现步骤**

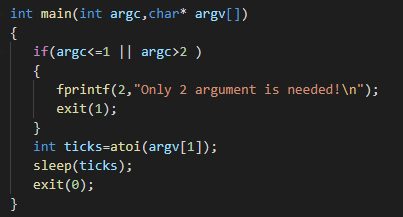
原代码已经在sys\_sleep系统调用中实现了sleep的功能，我们只需在user文件夹下创建一个sleep.c文件，并**结合传入的参数调用sys\_sleep系统调用**即可。此外，还要**在MakeFile文件的UPROGS中添加sleep选项**，当用户输入sleep命令后自动执行sleep.c的main函数。

具体的实现方法如下：

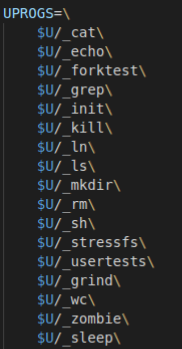
1. 创建sleep.c文件，并参考echo.c，引用头文件。



1. 编写sleep.c文件的main函数，具体内容如下：
   1. 判断输入参数个数是否符合要求。
   2. 调用atoi函数，将输入的字符串转换为int型变量。
   3. 调用sleep系统调用，进入休眠。
   4. 调用exit函数退出sleep逻辑。



3）在MakeFile文件UPROGS中添加sleep项，使用户输入sleep时，程序能执行sleep.c的main函数。



**3. 问题与解决方法**

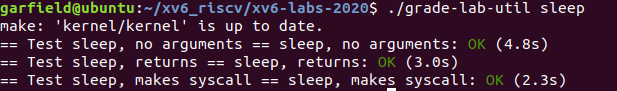
**问题一**：没有修改MAKEFILE文件，导致exec failed

**解决方法**：在MAKEFILE文件的PROC项中添加”$U/\_sleep\”项，保证用户输入sleep命令时会自动执行sleep.c的main函数。

**4. 结果**

1）输入sleep 10后，程序休眠一小会。

2）通过grade测试



**任务2 pingpong**

1. **问题描述**

本任务要求使用管道在父子进程间传输字节。需要注意的是，**管道逻辑上只能单向传输数据，但新创建的管道的两个读端和写端都是开放的，需要人为地关闭一个读端和一个写端。**

1. **思路与实现步骤**

首先创建2个管道，分别将其写端分配给父进程和子进程，然后父进程、子进程各向对方发送一个字节，并读取对方发送的字节。父进程先读后写，子进程先写后读。

具体的实现方法如下：

1. 创建pingpong.c文件，并参考echo.c，引用头文件（同任务1：sleep）。
2. 实现pingpong.c文件的main函数，具体内容如下：
   1. 判断命令行输入的参数个数是否符合要求。
   2. 创建2个管道，各关闭1个读端和1个写端，得到两个单向的管道。
   3. 创建子进程。
   4. 父进程先向第一个管道中写入1个字节，然后在另一个管道中阻塞等待读取子进程写入的1个字节。**（先写后读）**
   5. 子进程先在第一个管道中阻塞等待读取父进程写入的1个字节，然后在另一个管道中写入1个字节。**（先读后写）**
   6. 调用exit函数，退出pingpong逻辑。



1. 在MakeFile文件UPROGS中添加pingpong项，使用户输入pingpong时，程序能执行pingpong.c的main函数。（同任务1：sleep）
2. **问题与解决方法**

**问题一**：管道的逻辑描述与具体实现间的差距。

**解决方法**：

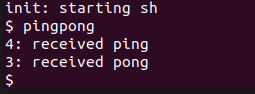
在逻辑意义上，管道是在进程间传输数据的工具，**只允许单向传输**。

在xv6系统中，**管道由指向同一文件的两个int型文件描述符构成**，通常将两个描述符组织成一个int型数组，数组0号元素只有读权限，1号元素只有写权限，二者分别是管道的读端和写端。父进程创建子进程后，两个进程均对文件有读、写权限，相当于持有管道的读端和写端。

为吻合管道的逻辑意义，也为了防止双向读写可能导致的**数据紊乱、竞争和死锁**，在xv6系统中**创建管道后要关闭管道的一个读端和一个写端**，使得一个进程只能写入数据，另一个进程只能读取数据，这样就得到了一个单向的管道。

**本任务**对管道的操作较为简单，**即使在创建管道后不关闭多余的读写端，也不会干扰程序的正常运行。**

1. **结果**
2. 输入pingpong后程序输出如下，符合要求



1. grade测试

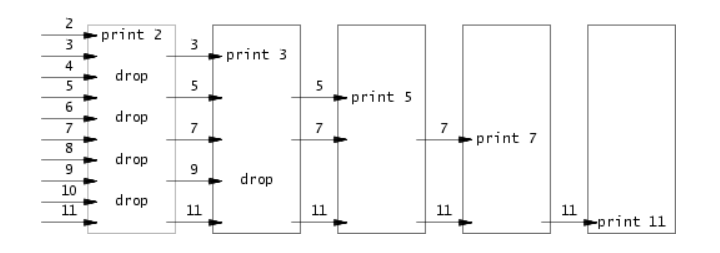


**任务3 primes**

1. **问题描述**

使用管道实现“质数筛”，能够自动筛选出2~35中的所有质数并打印到控制台，要求程序能够由多个进程并行地执行。

“质数筛”的原理如下图所示，每个进程从管道中读出剩余的数据，打印其中最小的整数（记作n），并剔除所有能被n整除的数，将剩余的数存入另一个管道中，供下一个进程筛选。只需要N-1个进程，就能够筛选出2~N范围内的所有质数。

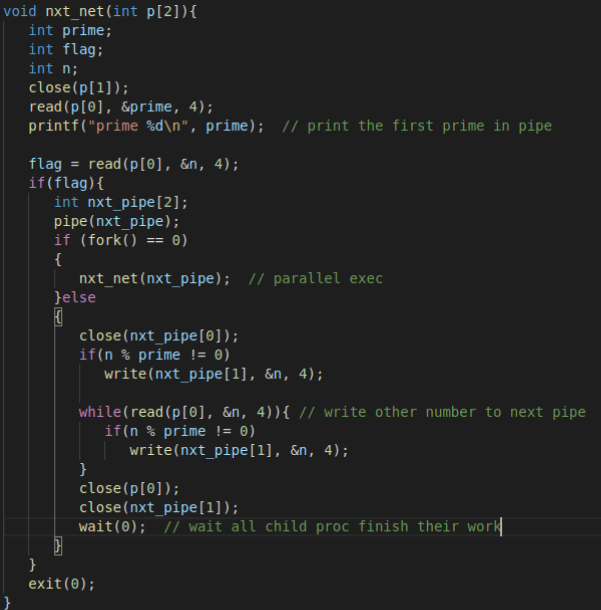


1. **思路与实现步骤**

基本思路已在“问题描述”中阐述，为实现多个进程并行地执行任务，设计了一个递归函数：首先读取并打印最小整数，然后创建一个子进程，由子进程运行下一级质数筛，父进程执行筛选剩余整数，存储不整除的整数等逻辑。

具体的实现方法如下所示：

1. 创建primes.c文件，并参考echo.c，引用头文件（同任务1：sleep）。
2. 实现primes.c文件的main函数，将数字2~35存入管道中，并调用递归函数。
3. 实现递归函数nxt\_net，具体内容如下：
   1. 读取管道中最小的（第一个）整数，并打印之，因为**管道中剩余的最小整数（记作n）通过了2~n-1的整数的筛选，不被其中的任何数整除，一定是质数**。
   2. 创建供下一级质数筛使用的管道。
   3. 创建子进程，父子进程执行不同的逻辑：
      1. 子进程递归地调用nxt\_net函数，首先阻塞等待管道中的数据，当读取到数据后开始执行下一级质数筛的逻辑。
      2. 父进程使用本级筛出的质数（管道中的第一个数）筛选管道中的整数，将不能整除的存入下一级管道，供下一级质数筛筛选。



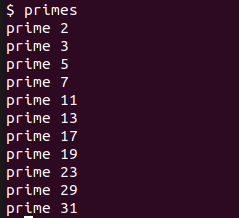
1. 在MakeFile文件UPROGS中添加primes项。（同任务1：sleep）
2. **问题与解决方法**

**问题一**：初次实现时，令递归函数中的父进程直接等待，退出，由子进程执行打印质数、筛选存储不整除的数、递归调用下一级质数筛等所有逻辑。**导致程序只能串行地执行，不满足题目中多进程并行工作的要求。**

**解决方法**：

经分析，**不能让“筛选不能整除的数”和“递归调用下一级质数筛”两个功能存在串行关系**，故**将筛选功能移动到父进程中实现**，使得以上两个任务能够在两个进程中并行地被执行。

1. **结果**
2. 输入primes后程序输出如下，符合要求



1. grade测试



**任务4 find**

1. **问题描述**

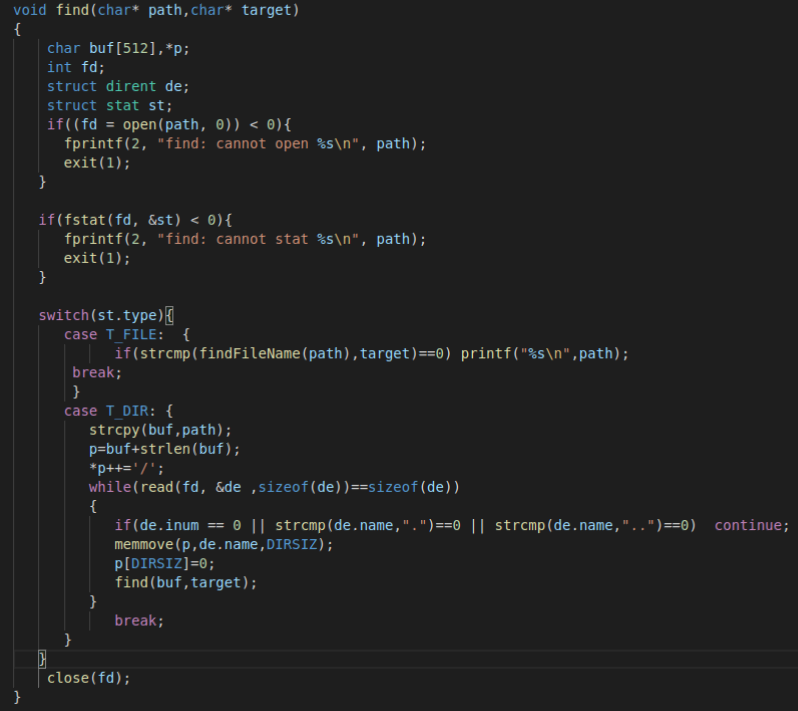
要求实现一个简易的文件查找程序，查找指定目录即各级子目录中具有指定名称的文件，并将结果输出到控制台中。

1. **思路与实现步骤**

要递归地查找子目录，可以实现一个递归函数，遍历目录中的所有文件（和文件夹），判断各文件的名称是否符合要求，并递归地在子文件夹中继续查找。

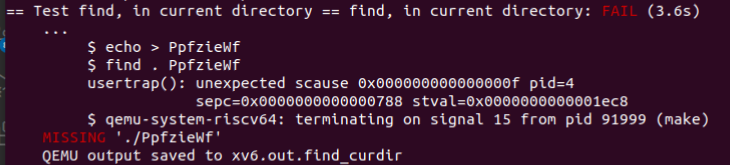
具体实现方法如下：

1. 创建find.c文件，并参考echo.c，引用头文件。（同任务1：sleep）
2. 实现find.c文件的main函数，从命令行输入中读取目标文件夹、目标文件名等信息，并调用递归查找函数。
3. 实现findFileName函数，给定文件路径，根据”/”从中分割出文件名。
4. 编写递归查找函数，具体内容如下：
   1. 使用open函数打开目标路径。
   2. 使用fstat函数获取目标文件（文件夹）的stat信息，其中包含了文件的类型(stat.type)是文件还是文件夹。
   3. 对不同类型的目标文件（文件夹）执行不同的操作：
      1. 若目标文件不是文件夹，则调用fileFindName函数和strcmp函数判断该文件的文件名与target文件名是否一致，若一致则打印该文件的路径。
      2. 若目标文件是文件夹，则分情况讨论：
         1. 若目标文件夹中没有文件，或文件夹名称为”.”（当前文件夹）或”..”（上一级文件夹），则跳过该文件夹。
         2. 否则递归调用查找函数，在子文件夹中继续查找符合要求的文件。



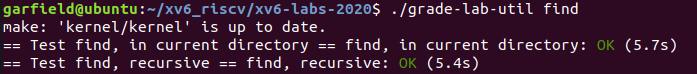
1. 在MakeFile文件UPROGS中添加find项。（同任务1：sleep）
2. **问题与解决方法**

**问题一**：初次实现时没有考虑**”.”、”..”等有特殊意义的文件夹名称**，导致不能通过make grade。



**解决方法**：添加了条件判断语句，**跳过特殊文件夹**。

1. **结果**
2. grade测试



**任务5 xargs**

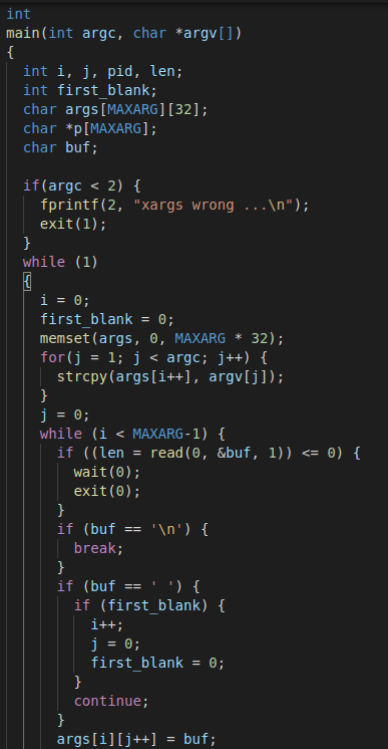
**1. 问题描述**

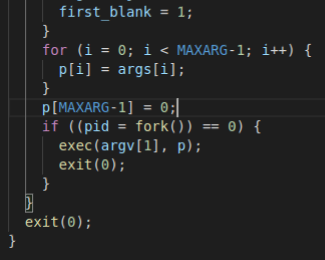
Xargs的格式为 aaa|bbb c，其中aaa为一个命令，xargs会将aaa命令的执行结果作为参数输入竖线右侧，实现从输出到输入的转化。

要求在xv6系统中实现一个简易的xargs命令。

**2. 思路与实现步骤**

1. 创建xargs.c文件，并参考echo.c，引用头文件。（同任务1：sleep）
2. 实现main函数，具体内容为：
   1. 从命令行中读取参数，并判断第一个参数是否合理。
   2. 第二个参数存储竖线右侧的字符串（命令）。
   3. **以换行符为终止符，空格为间隔符**，将竖线左侧命令的执行结果切割为char\*型数组”p”。
   4. **调用exec命令，将竖线左侧命令（argv[1]）与竖线左侧参数数组拼接在一起，作为1个字符串命令执行**。





1. 在MakeFile文件UPROGS中添加xargs项。（同任务1：sleep）

**3. 问题与解决方法**

**问题一**：实现本任务时不清楚xargs的具体功能。

**解决方法**：在菜鸟教程上查阅了Linux系统中xargs的功能文档，理解了xargs的功能。

**4. 结果**

1. grade测试



**四、实验结果**

通过make grade测试。

