项目说明文档

操作系统课程设计

——xv6 及 Labs 课程项目

作 者 姓 名： 袁泓博

学 号： 2152208

指 导 教 师： 王冬青

学 院 专 业： 软件学院 软件工程

同济大学

Tongji University

目录

[**1.** **Lab1: Xv6 and Unix utilities** 6](#_Toc142915040)

[1.1 Boot xv6 (easy) 6](#_Toc142915041)

[1.1.1 实验目的 6](#_Toc142915042)

[1.1.2 实验步骤 6](#_Toc142915043)

[1.1.3 实验中遇到的问题和解决办法 6](#_Toc142915044)

[1.1.4 实验心得 6](#_Toc142915045)

[1.2 sleep (easy) 6](#_Toc142915046)

[1.2.1 实验目的 6](#_Toc142915047)

[1.2.2 实验步骤 6](#_Toc142915048)

[1.2.3 实验中遇到的问题和解决办法 8](#_Toc142915049)

[1.2.4 实验心得 8](#_Toc142915050)

[1.3 pingpong (easy) 8](#_Toc142915051)

[1.3.1 实验目的 8](#_Toc142915052)

[1.3.2 实验步骤 8](#_Toc142915053)

[1.3.3 实验中遇到的问题和解决办法 9](#_Toc142915054)

[1.3.4 实验心得 10](#_Toc142915055)

[1.4 primes (moderate)/(hard) 10](#_Toc142915056)

[1.4.1 实验目的 10](#_Toc142915057)

[1.4.2 实验步骤 10](#_Toc142915058)

[1.4.3 实验中遇到的问题和解决办法 11](#_Toc142915059)

[1.4.4 实验心得 11](#_Toc142915060)

[1.5 find (moderate) 12](#_Toc142915061)

[1.5.1 实验目的 12](#_Toc142915062)

[1.5.2 实验步骤 12](#_Toc142915063)

[1.5.3 实验中遇到的问题和解决办法 13](#_Toc142915064)

[1.5.4 实验心得 13](#_Toc142915065)

[1.6 xargs (moderate) 14](#_Toc142915066)

[1.6.1 实验目的 14](#_Toc142915067)

[1.6.2 实验步骤 14](#_Toc142915068)

[1.6.3 实验中遇到的问题和解决办法 16](#_Toc142915069)

[1.6.4 实验心得 16](#_Toc142915070)

[**2.** **Lab2: system calls** 17](#_Toc142915071)

[2.1 System call tracing (moderate) 17](#_Toc142915072)

[2.1.1 实验目的 17](#_Toc142915073)

[2.1.2 实验步骤 17](#_Toc142915074)

[2.1.3 实验中遇到的问题和解决办法 20](#_Toc142915075)

[2.1.4 实验心得 20](#_Toc142915076)

[2.2 Sysinfo (moderate) 20](#_Toc142915077)

[2.2.1 实验目的 20](#_Toc142915078)

[2.2.2 实验步骤 20](#_Toc142915079)

[2.2.3 实验中遇到的问题和解决办法 23](#_Toc142915080)

[2.2.4 实验心得 23](#_Toc142915081)

[**3.** **Lab3: page tables** 23](#_Toc142915082)

[3.1 Speed up system calls (easy) 23](#_Toc142915083)

[3.1.1 实验目的 23](#_Toc142915084)

[3.1.2 实验步骤 23](#_Toc142915085)

[3.1.3 实验中遇到的问题和解决办法 23](#_Toc142915086)

[3.1.4 实验心得 23](#_Toc142915087)

[3.2 Print a page table (easy) 24](#_Toc142915088)

[3.2.1 实验目的 24](#_Toc142915089)

[3.2.2 实验步骤 24](#_Toc142915090)

[3.2.3 实验中遇到的问题和解决办法 24](#_Toc142915091)

[3.2.4 实验心得 24](#_Toc142915092)

[3.3 Detecting which pages have been accessed (hard) 24](#_Toc142915093)

[3.3.1 实验目的 24](#_Toc142915094)

[3.3.2 实验步骤 24](#_Toc142915095)

[3.3.3 实验中遇到的问题和解决办法 24](#_Toc142915096)

[3.3.4 实验心得 24](#_Toc142915097)

[**4.** **Lab4: traps** 24](#_Toc142915098)

[4.1 RISC-V assembly (easy) 24](#_Toc142915099)

[4.1.1 实验目的 24](#_Toc142915100)

[4.1.2 实验步骤 24](#_Toc142915101)

[4.1.3 实验中遇到的问题和解决办法 24](#_Toc142915102)

[4.1.4 实验心得 25](#_Toc142915103)

[4.2 Backtrace (moderate) 25](#_Toc142915104)

[4.2.1 实验目的 25](#_Toc142915105)

[4.2.2 实验步骤 25](#_Toc142915106)

[4.2.3 实验中遇到的问题和解决办法 25](#_Toc142915107)

[4.2.4 实验心得 25](#_Toc142915108)

[4.3 Alarm (hard) 25](#_Toc142915109)

[4.3.1 实验目的 25](#_Toc142915110)

[4.3.2 实验步骤 25](#_Toc142915111)

[4.3.3 实验中遇到的问题和解决办法 25](#_Toc142915112)

[4.3.4 实验心得 25](#_Toc142915113)

[**5.** **Lab5: Copy-on-Write Fork for xv6** 26](#_Toc142915114)

[5.1 Implement copy-on write(hard) 26](#_Toc142915115)

[5.1.1 实验目的 26](#_Toc142915116)

[5.1.2 实验步骤 26](#_Toc142915117)

[5.1.3 实验中遇到的问题和解决办法 26](#_Toc142915118)

[5.1.4 实验心得 26](#_Toc142915119)

[**6.** **Lab6: Multithreading** 26](#_Toc142915120)

[6.1 Uthread: switching between threads (moderate) 26](#_Toc142915121)

[6.1.1 实验目的 26](#_Toc142915122)

[6.1.2 实验步骤 26](#_Toc142915123)

[6.1.3 实验中遇到的问题和解决办法 26](#_Toc142915124)

[6.1.4 实验心得 26](#_Toc142915125)

[6.2 Using threads (moderate) 27](#_Toc142915126)

[6.2.1 实验目的 27](#_Toc142915127)

[6.2.2 实验步骤 27](#_Toc142915128)

[6.2.3 实验中遇到的问题和解决办法 27](#_Toc142915129)

[6.2.4 实验心得 27](#_Toc142915130)

[6.3 Barrier(moderate) 27](#_Toc142915131)

[6.3.1 实验目的 27](#_Toc142915132)

[6.3.2 实验步骤 27](#_Toc142915133)

[6.3.3 实验中遇到的问题和解决办法 27](#_Toc142915134)

[6.3.4 实验心得 27](#_Toc142915135)

[**7.** **Lab7: networking** 27](#_Toc142915136)

[7.1 Your Job (hard) 27](#_Toc142915137)

[7.1.1 实验目的 27](#_Toc142915138)

[7.1.2 实验步骤 27](#_Toc142915139)

[7.1.3 实验中遇到的问题和解决办法 27](#_Toc142915140)

[7.1.4 实验心得 28](#_Toc142915141)

[**8.** **Lab8: locks** 28](#_Toc142915142)

[8.1 Memory allocator (moderate) 28](#_Toc142915143)

[8.1.1 实验目的 28](#_Toc142915144)

[8.1.2 实验步骤 28](#_Toc142915145)

[8.1.3 实验中遇到的问题和解决办法 28](#_Toc142915146)

[8.1.4 实验心得 28](#_Toc142915147)

[8.2 Buffer cache (hard) 28](#_Toc142915148)

[8.2.1 实验目的 28](#_Toc142915149)

[8.2.2 实验步骤 28](#_Toc142915150)

[8.2.3 实验中遇到的问题和解决办法 28](#_Toc142915151)

[8.2.4 实验心得 28](#_Toc142915152)

[**9.** **Lab9: file system** 29](#_Toc142915153)

[9.1 Large files (moderate) 29](#_Toc142915154)

[9.1.1 实验目的 29](#_Toc142915155)

[9.1.2 实验步骤 29](#_Toc142915156)

[9.1.3 实验中遇到的问题和解决办法 29](#_Toc142915157)

[9.1.4 实验心得 29](#_Toc142915158)

[9.2 Symbolic links (moderate) 29](#_Toc142915159)

[9.2.1 实验目的 29](#_Toc142915160)

[9.2.2 实验步骤 29](#_Toc142915161)

[9.2.3 实验中遇到的问题和解决办法 29](#_Toc142915162)

[9.2.4 实验心得 29](#_Toc142915163)

[**10.** **Lab10: mmap** 30](#_Toc142915164)

[10.1 mmap (hard) 30](#_Toc142915165)

[10.1.1 实验目的 30](#_Toc142915166)

[10.1.2 实验步骤 30](#_Toc142915167)

[10.1.3 实验中遇到的问题和解决办法 30](#_Toc142915168)

[10.1.4 实验心得 30](#_Toc142915169)

# **Lab1: Xv6 and Unix utilities**

## Boot xv6 (easy)

## 实验目的

启动xv6，初步掌握xv6

## 实验步骤

获取实验室的xv6源代码并切换到util分支：

$ **git clone git://g.csail.mit.edu/xv6-labs-2021**

$ **cd xv6-labs-2021**

$ **git checkout util**

利用make qemu指令运行xv6：

$ **make qemu**

此时输入ls指令能看到内容输出

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## sleep (easy)

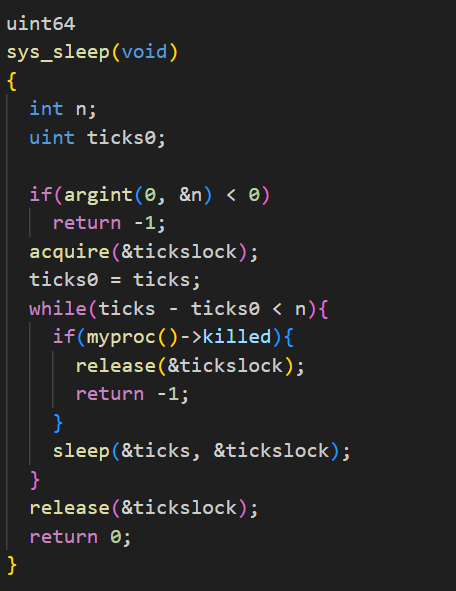
## 实验目的

实现xv6的UNIX程序sleep：您的sleep应该暂停到用户指定的计时数。

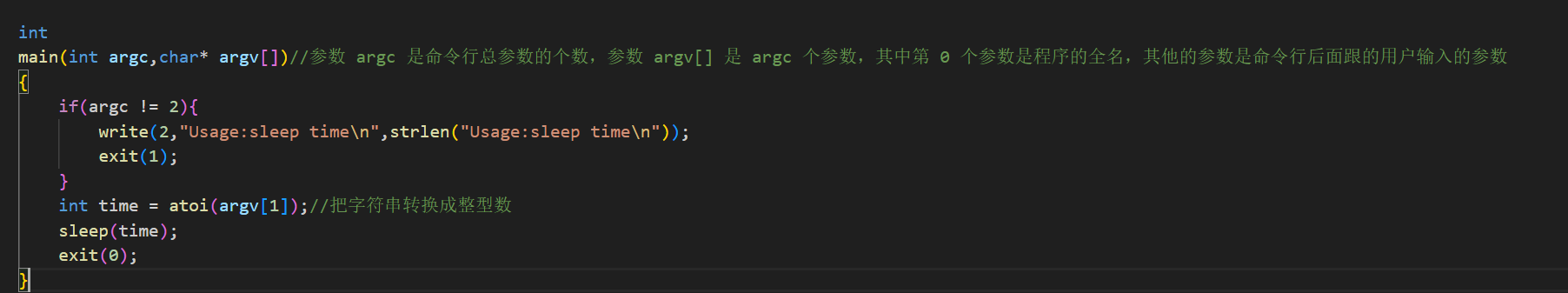
## 实验步骤

使用系统调用sleep

参阅kernel/sysproc.c以获取实现sleep系统调用的xv6内核代码：



user/user.h提供了sleep的声明以便其他程序调用，用汇编程序编写的user/usys.S可以帮助sleep从用户区跳转到内核区



确保main函数调用exit()以退出程序

将sleep程序添加到Makefile中的UPROGS中即可运行



## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## pingpong (easy)

## 实验目的

编写一个使用UNIX系统调用的程序来在两个进程之间“ping-pong”一个字节，使用两个管道，每个方向一个。父进程应该向子进程发送一个字节;子进程应该打印“<pid>: received ping”，其中<pid>是进程ID，并在管道中写入字节发送给父进程，然后退出;父级应该从读取从子进程而来的字节，打印“<pid>: received pong”，然后退出。

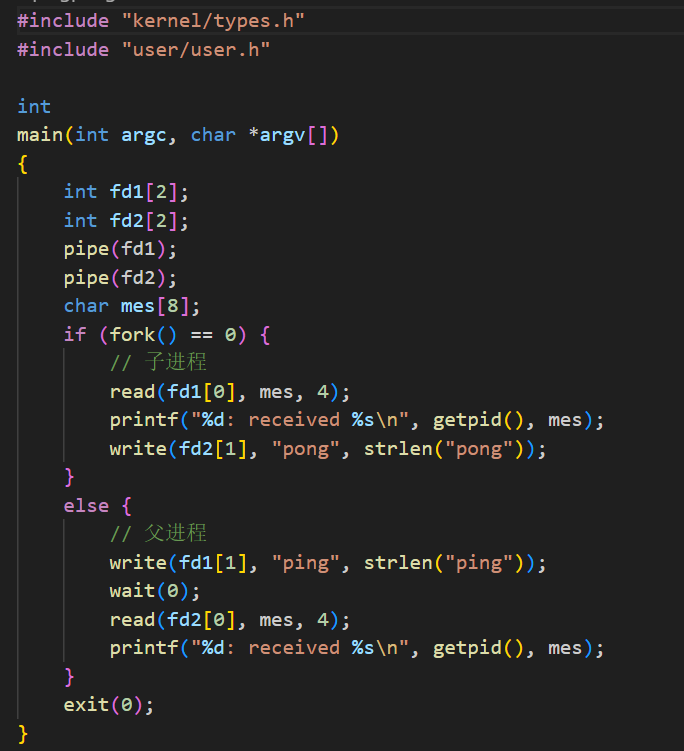
## 实验步骤

使用pipe来创造管道

使用fork创建子进程

使用read从管道中读取数据，并且使用write向管道中写入数据

使用getpid获取调用进程的pid：



将程序加入到Makefile的UPROGS



## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## primes (moderate)/(hard)

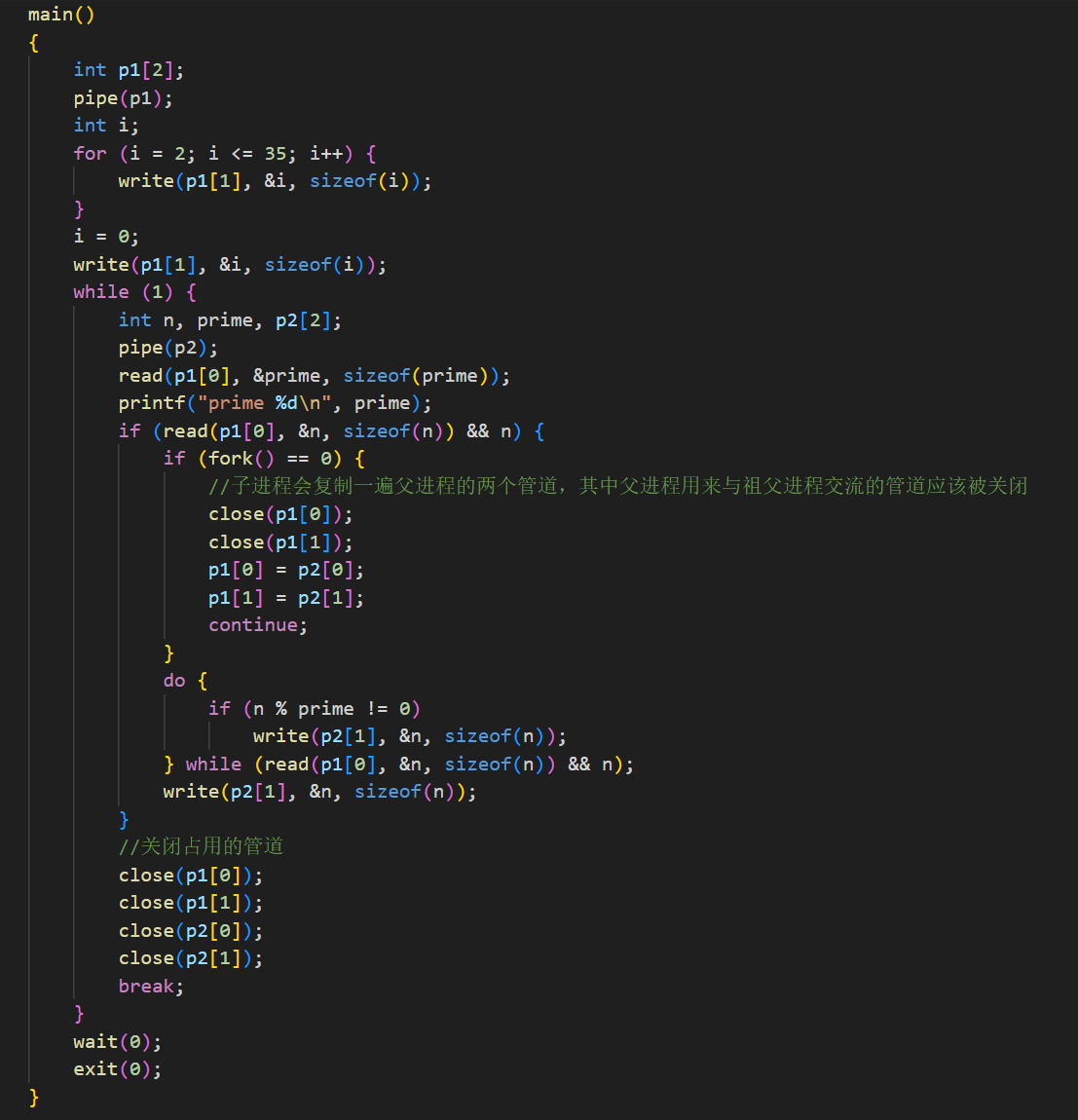
## 实验目的

使用管道编写prime sieve(筛选素数)的并发版本。

## 实验步骤

关闭进程不需要的文件描述符

一旦第一个进程达到35，它应该使用wait等待整个管道终止，包括所有子孙进程等等。因此，主primes进程应该只在打印完所有输出之后，并且在所有其他primes进程退出之后退出，当管道的write端关闭时，read返回零



将程序添加到Makefile中的UPROGS



## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

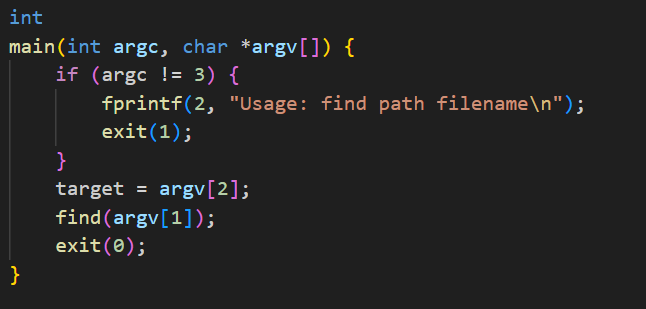
## find (moderate)

## 实验目的

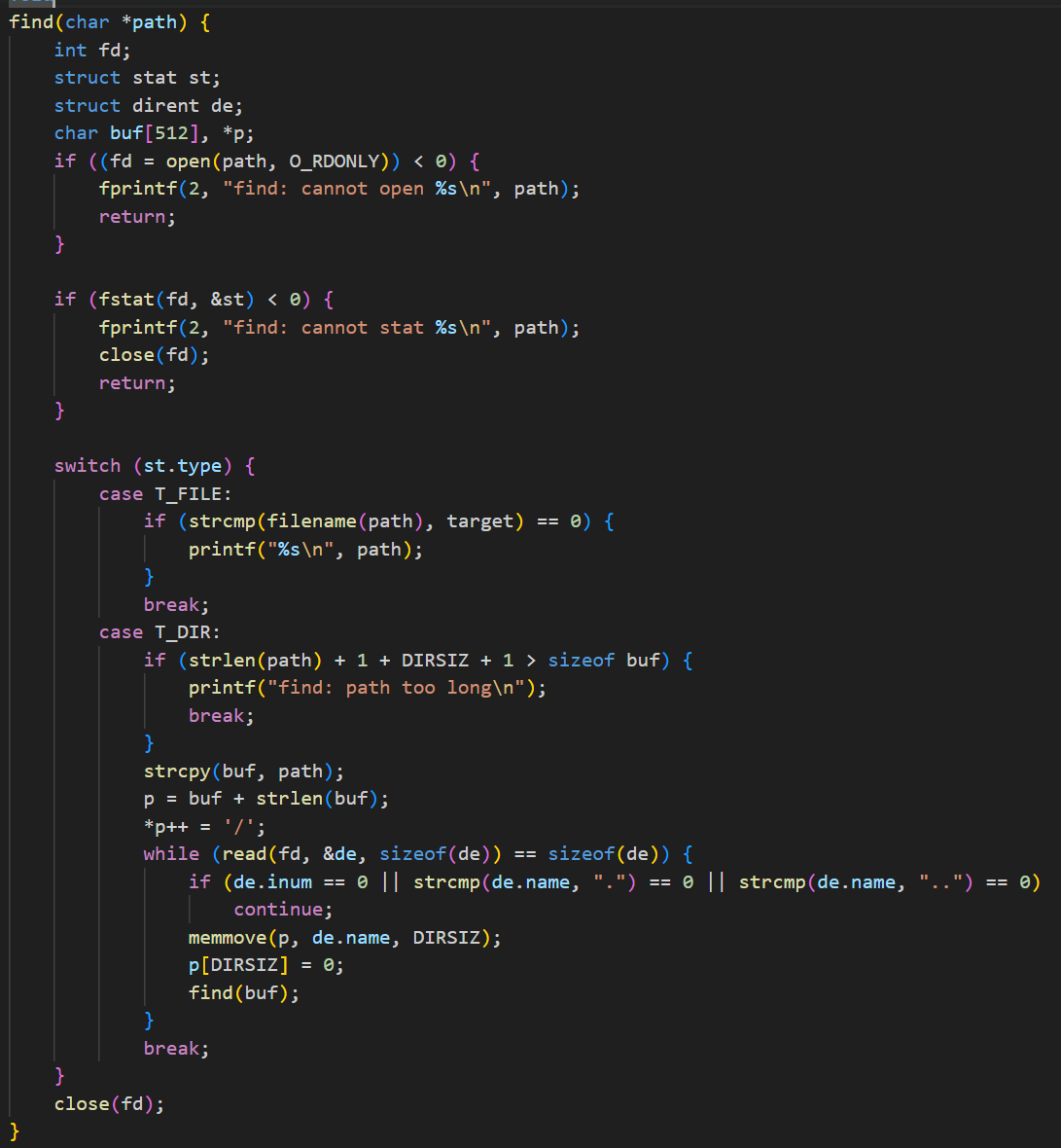
写一个简化版本的UNIX的find程序：查找目录树中具有特定名称的所有文件

## 实验步骤

参考 ls 方法，我可以发现 find 实现方法的整体思路应当为深度优先搜索。因此，在 main 函数中我只需要检查参数。如果参数合法，只需要直接调用深度优先搜索函数 find 即可：



对于 find 函数，它的作用是在每一级中检索是否有满足要求的文件：



在 MakeFile 文件中添加 find 方法即可



## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

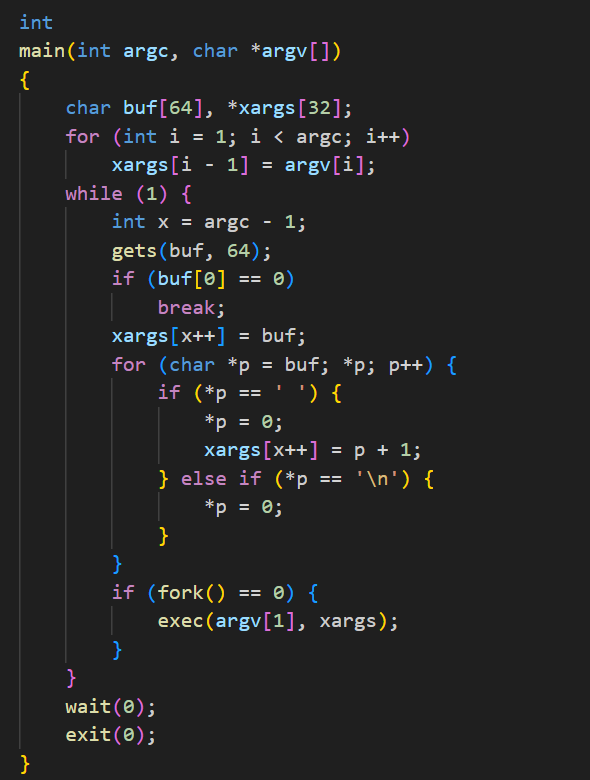
## xargs (moderate)

## 实验目的

编写一个简化版UNIX的xargs程序：它从标准输入中按行读取，并且为每一行执行一个命令，将行作为参数提供给命令

## 实验步骤

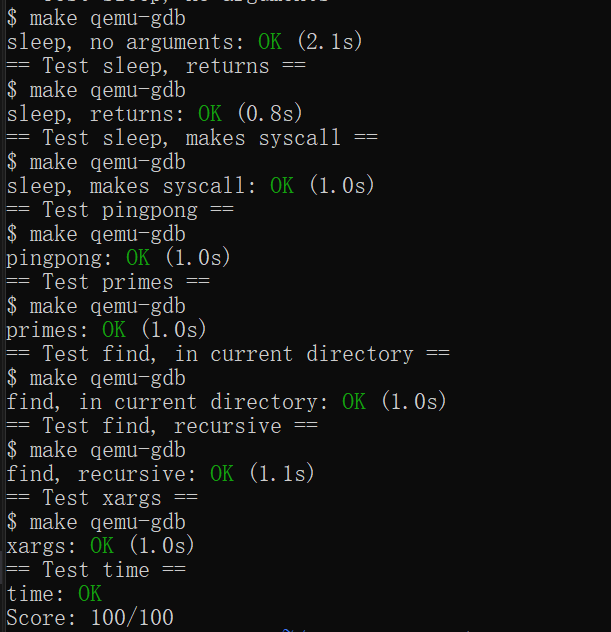
使用fork和exec对每行输入调用命令，在父进程中使用wait等待子进程完成命令。要读取单个输入行，一次读取一个字符，直到出现换行符（'\n'）



将程序添加到Makefile中的UPROGS：



最后所有实验检验得分：



## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

# **Lab2: system calls**

## System call tracing (moderate)

## 实验目的

通过向xv6添加一些新的系统调用，帮助了解系统调用是如何工作的，并了解xv6内核的一些内部结构。

## 实验步骤

将代码切换到syscall分支：

$ **git fetch**

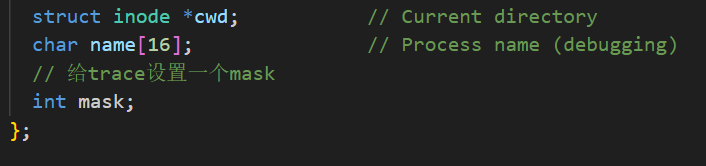
$ **git checkout syscall**

$ **make clean**

把Makefile加上：



我们要在/kernel/proc.h文件的proc结构体中加入一个新的变量，让每个进程都有一个自己的mask：



主要的实现就是在/usr/syscall.c文件的syscall函数，通过观察该函数可以发现：

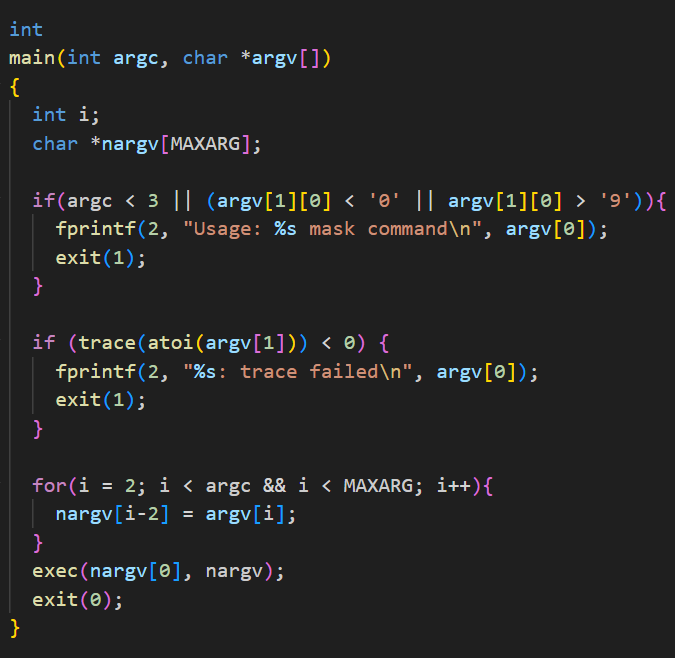
p->trapframe->a0 = syscalls[num]();

调用了系统调用命令，并且把返回值保存在了a0寄存器中（RISCV的C规范是把返回值放在a0中)，所以我们只要在调用系统调用时判断是不是mask规定的输出函数，如果是就输出。

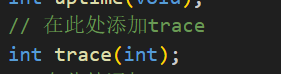
由于mask是按位判断的，且proc结构体里的name是整个线程的名字，不是函数调用的函数名称，所以我们不能用p->name，而要自己定义一个数组，因此最终函数写为：



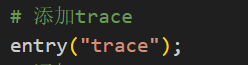
为了把mask参数传进来需要用到trace.c 文件，而这里已经给了trace的用户态函数：



可以看到trace函数传入的是一个数字，那我们只要在系统调用trace里把这个数字给到现在的线程，把trace这个系统调用加入到内核中声明，首先是/usre/user.h文件加入，这里声明了用户态可以调用的系统调用：



/user/usys.pl文件加入：



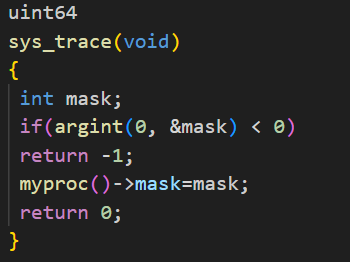
接下来还要给内核态的系统调用trace加上声明和定义，在kernel/syscall.c加上：



在下面的函数指针数组\*syscalls[]加上：



然后在kerlnel/sysproc.c加上sys\_trace的定义实现：



即可成功运行

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Sysinfo (moderate)

## 实验目的

添加系统调用 sysinfo，用于收集正在运行的系统的信息。

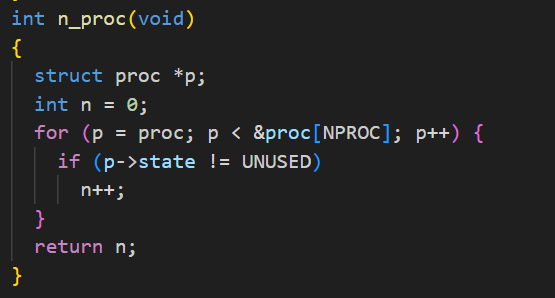
## 实验步骤

通过阅读发现在/kernel/proc.c文件中定义了一个数组：

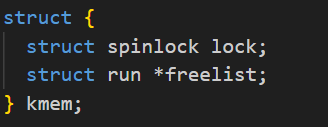


这个数组就保存着所有的进程，所以只要遍历这个数组判断状态就行

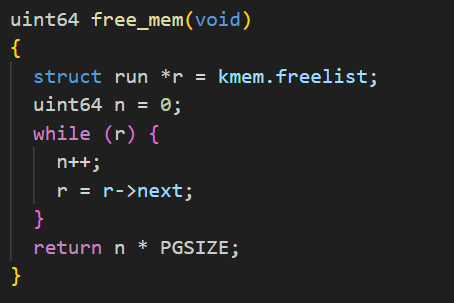
所以实现在kernel/proc.c中加入：



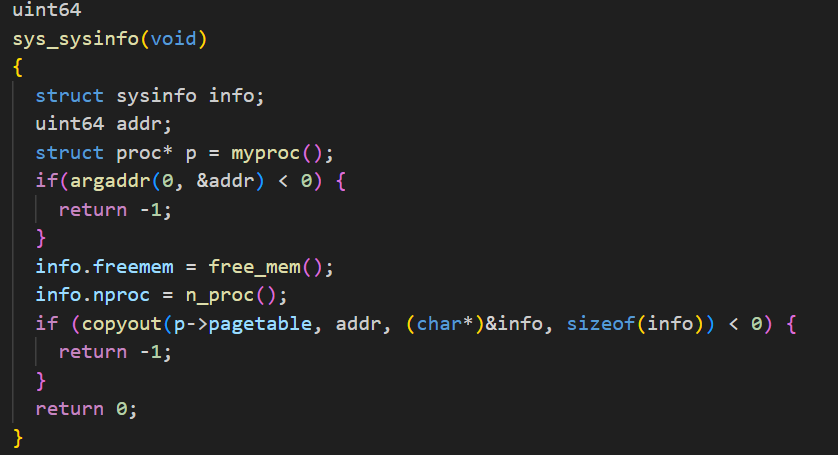
针对于当前可以使用的空间，可以参考 kalloc.c 文件。其中定义了一个链表 kemem，每个链表都指向了一个可用空间，而 kmem 则保存了最后一个链表



所以kmem.freelist永远指向最后一个可用页，那我们只要顺着这个链表往前走，直到NULL为止：

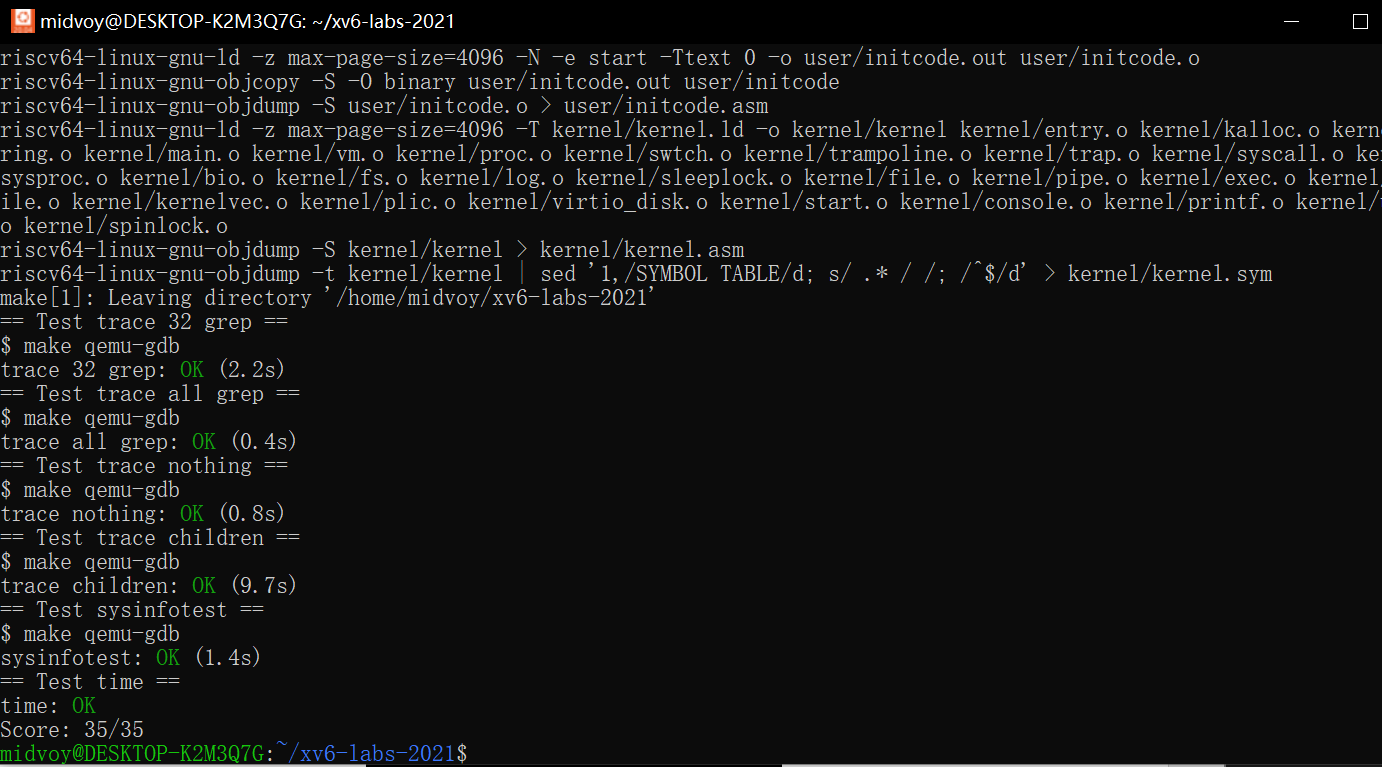


然后在kernel/sysproc.c文件里加入sysinfo实现，同时加入sysinfo结构体对应头文件：



即完成本次实验。

通过make grade测试：



## 实验中遇到的问题和解决办法

[待填]

## 实验心得

待填

# **Lab3: page tables**

## Speed up system calls (easy)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Print a page table (easy)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Detecting which pages have been accessed (hard)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

# **Lab4: traps**

## RISC-V assembly (easy)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Backtrace (moderate)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Alarm (hard)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

# **Lab5: Copy-on-Write Fork for xv6**

## Implement copy-on write(hard)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

# **Lab6: Multithreading**

## Uthread: switching between threads (moderate)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Using threads (moderate)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Barrier(moderate)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

# **Lab7: networking**

## Your Job (hard)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

# **Lab8: locks**

## Memory allocator (moderate)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Buffer cache (hard)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

# **Lab9: file system**

## Large files (moderate)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

## Symbolic links (moderate)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填

# **Lab10: mmap**

## mmap (hard)

## 实验目的

## 实验步骤

## 实验中遇到的问题和解决办法

待填

## 实验心得

待填