Lab：Xv6 and Unix utilities

所有实验均在虚拟机环境，Ubuntu操作系统中进行。本项实验由六部分组成，实验顺序依次是Boot xv6、sleep、pingpong、primes、find、xargs。

1.Boot xv6

实验目的：

利用 QEMU 模拟器启动 xv6 系统。

实验步骤：

1. 安装必要的软件：QEMU、GCC、GDB 等

2.克隆 xv6 源码库：

git clone git://g.csail.mit.edu/xv6-labs-2020

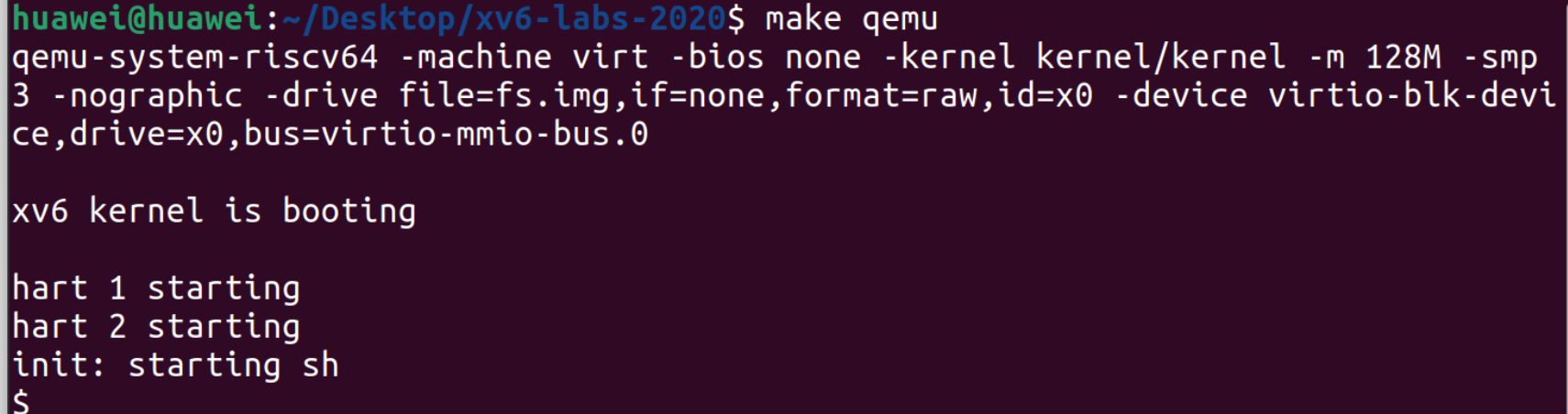
cd xv6-labs-2020

3.编译和运行 xv6：

make

make qemu

实验结果与实验心得

实验结果：

该实验是后续实验的基础，是一个非常简单的操作实验。

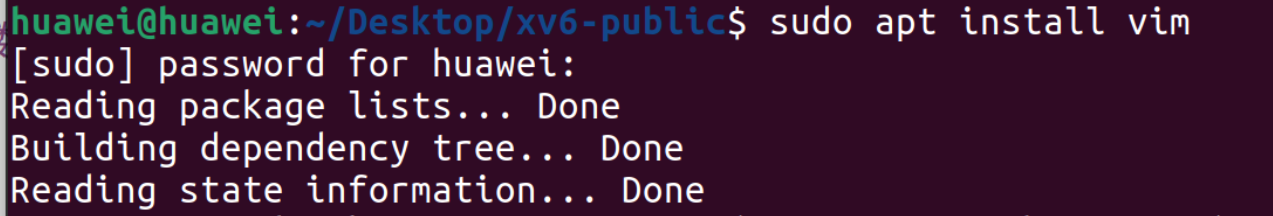
2.Sleep

实验目的

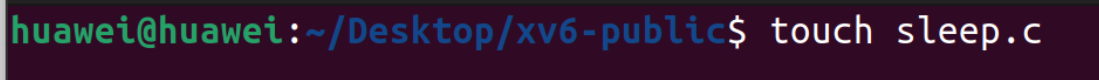
为 xv6 系统实现 UNIX 的 sleep 程序。sleep 程序使得当前进程暂停一定的时钟周期数，时钟周期数由用户指定。例如 sleep 100 ，则当前进程暂停，等待 100 个时钟周期后才继续执行。

实验步骤

1.先使用命令 sudo apt install vim 安装 vim 编辑器。



2.在文件目录下创建sleep.c。



3.编写sleep.c文件使得程序符合要求。

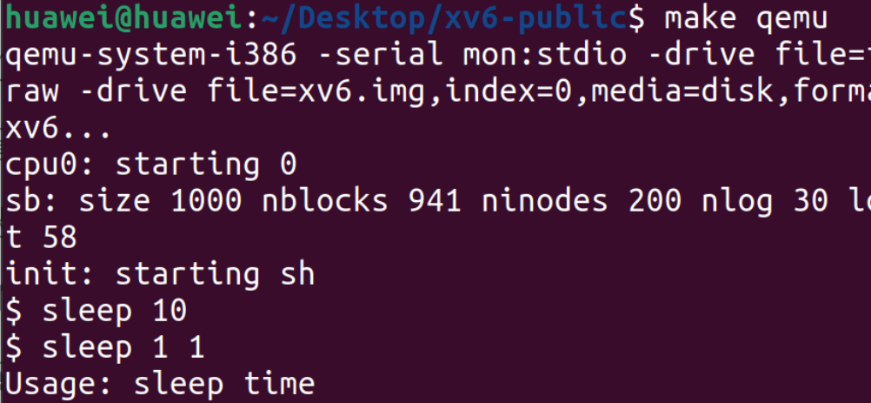


4. 在 Makefile 文件中添加配置：\_sleep\。



实验结果与实验心得

编译运行结果如下，实验成功。



起初编写程序时，使用了现有的框架，后来编译运行时出现错误：exit（）的调用错误。后来通过查看错误信息和user.h文件，修改了文件，才得以通过。

3.Pingpong

实验目的

使用 UNIX 系统调用编写一个程序 pingpong ，在一对管道上实现两个进程之间的通信。父进程应该通过第一个管道给子进程发送一个信息 “ping”，子进程接收父进程的信息后打印 "<pid>: received ping" ，其中<pid>是其进程 ID 。然后子进程通过另一个管道发送一个信息 “pong” 给父进程，父进程接收子进程的信息然后打印 "<pid>: received pong" ，然后退出。

实验步骤

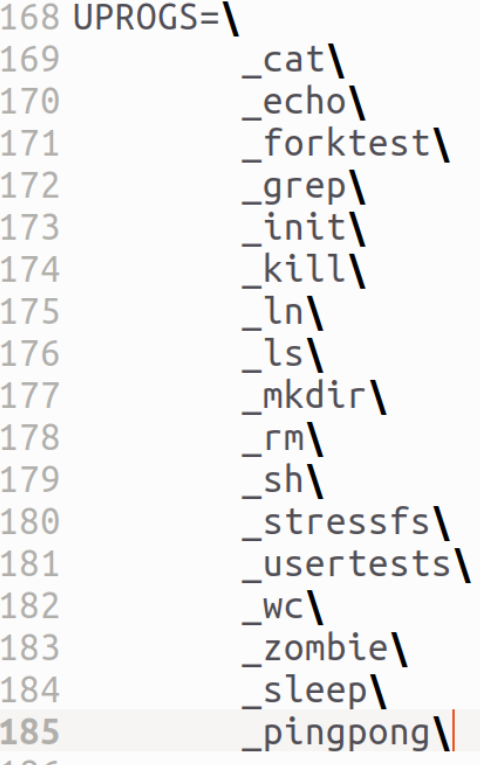
1.创建pingpong.c文件



2.编写程序



3.在 Makefile 文件中添加配置：\_pingpong\。



实验结果与心得

实验结果如下，表明实验成功。



在编写程序前要比较清楚系统提供的函数，比如在该实验中，pipe（）函数用于定义进行通信的管道，其中一个为读，一个为写。而fork（）函数用于进程的生成，新创建的子进程会拥有自己独立的进程标识符（PID），父子进程拥有不同PID，可以利用这个特性来区分。对于父进程，`fork`会返回子进程的PID，而对于子进程，`fork`会返回0。

4.Primes

实验目的

使用管道将 2 至 35 中的素数筛选出来，这个想法归功于 Unix 管道的发明者 Doug McIlroy 。链接中间的图片和周围的文字解释了如何操作。最后的解决方案应该放在 user/primes.c 文件中。

你的目标是使用 pipe 和 fork 来创建管道。第一个进程将数字 2 到 35 送入管道中。对于每个质数，你要安排创建一个进程，从其左邻通过管道读取，并在另一条管道上写给右邻。由于 xv6 的文件描述符和进程数量有限，第一个进程可以停止在 35 。



实验步骤

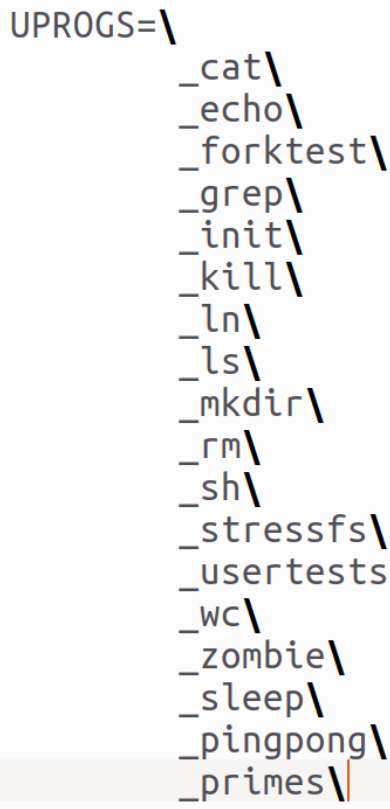
1.创建primes.c文件



2.编写程序

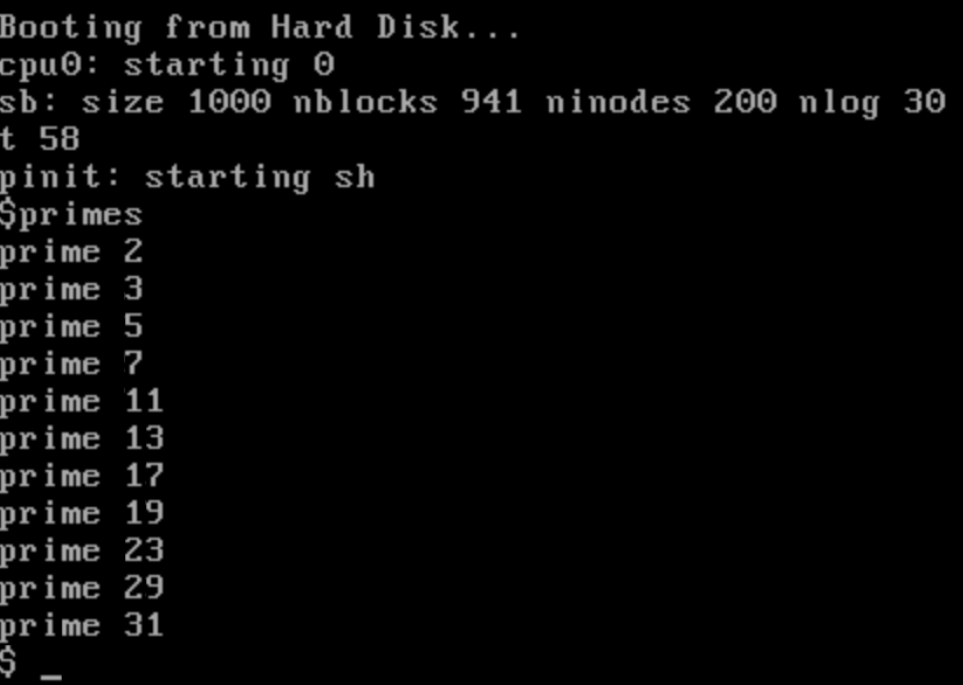


3.在 Makefile 文件中添加配置：\_primes\。



实验结果与心得

实验结果如下，表明实验成功。

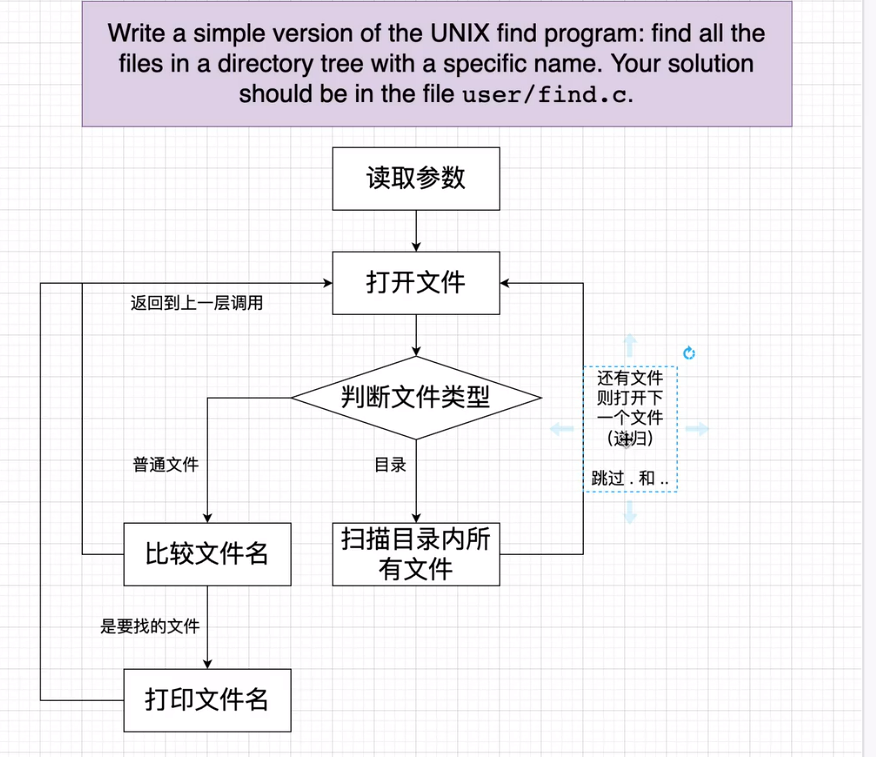


管道用于进程间通信，在编写过程中必须小心管理管道的读端和写端的打开和关闭，以防止资源泄漏或死锁。进程的创建和管理也需要谨慎处理，确保父进程和子进程之间的同步。递归调用 sieve 函数，每次筛选出当前素数并传递剩余数字，这需要确保管道和进程在递归调用中的正确初始化和关闭。该实验并不算太难，主要是要理解管道和父子进程的创建管理。

5.Find

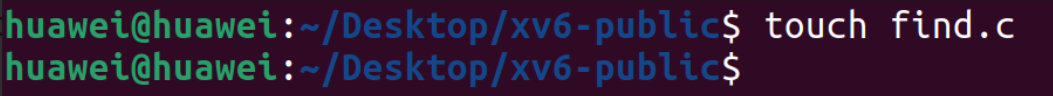
实验目的

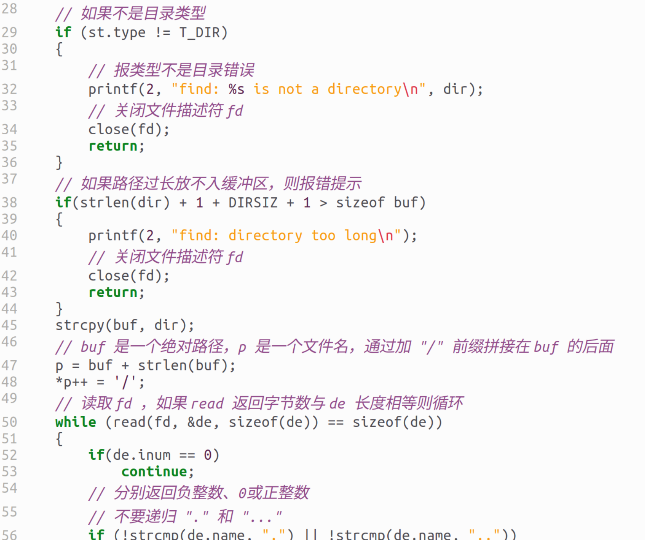
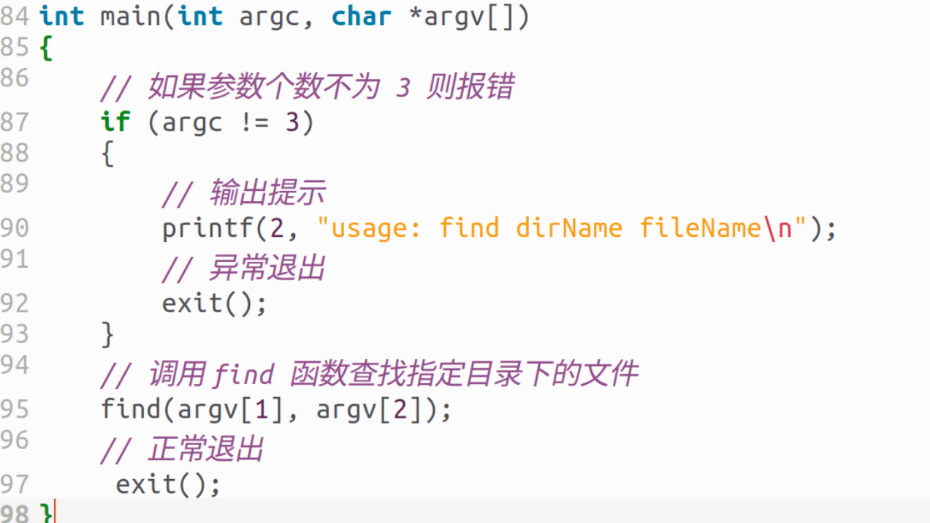
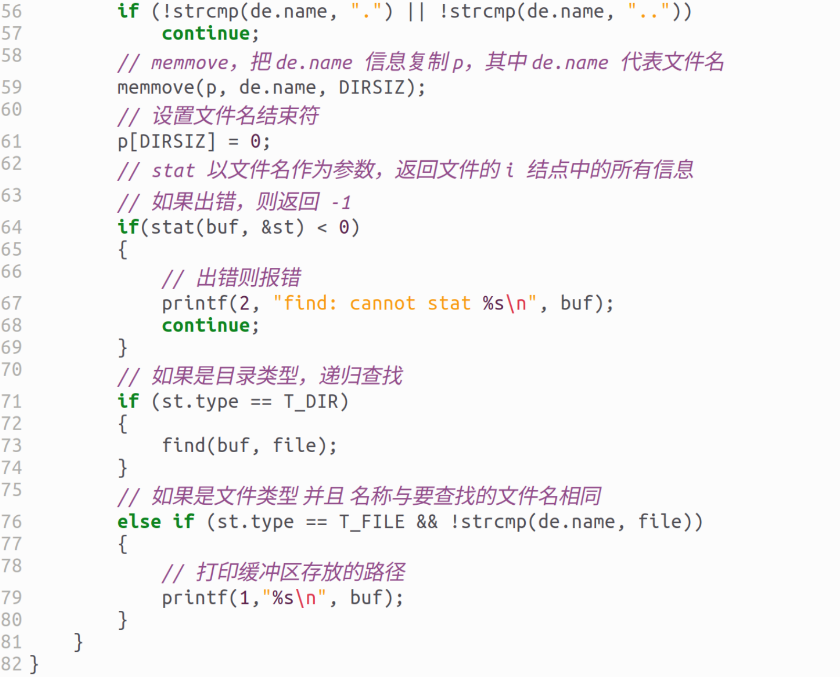
编写一个简单的 UNIX find 程序，在目录树中查找包含特定名称的所有文件。



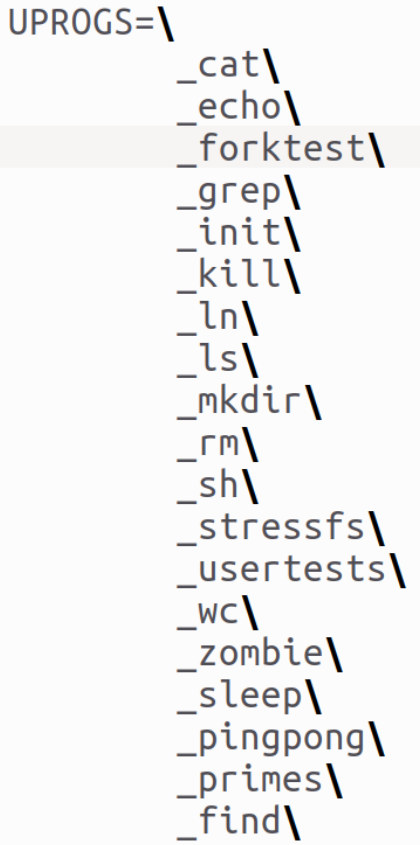
实验步骤

1.创建find.c文件

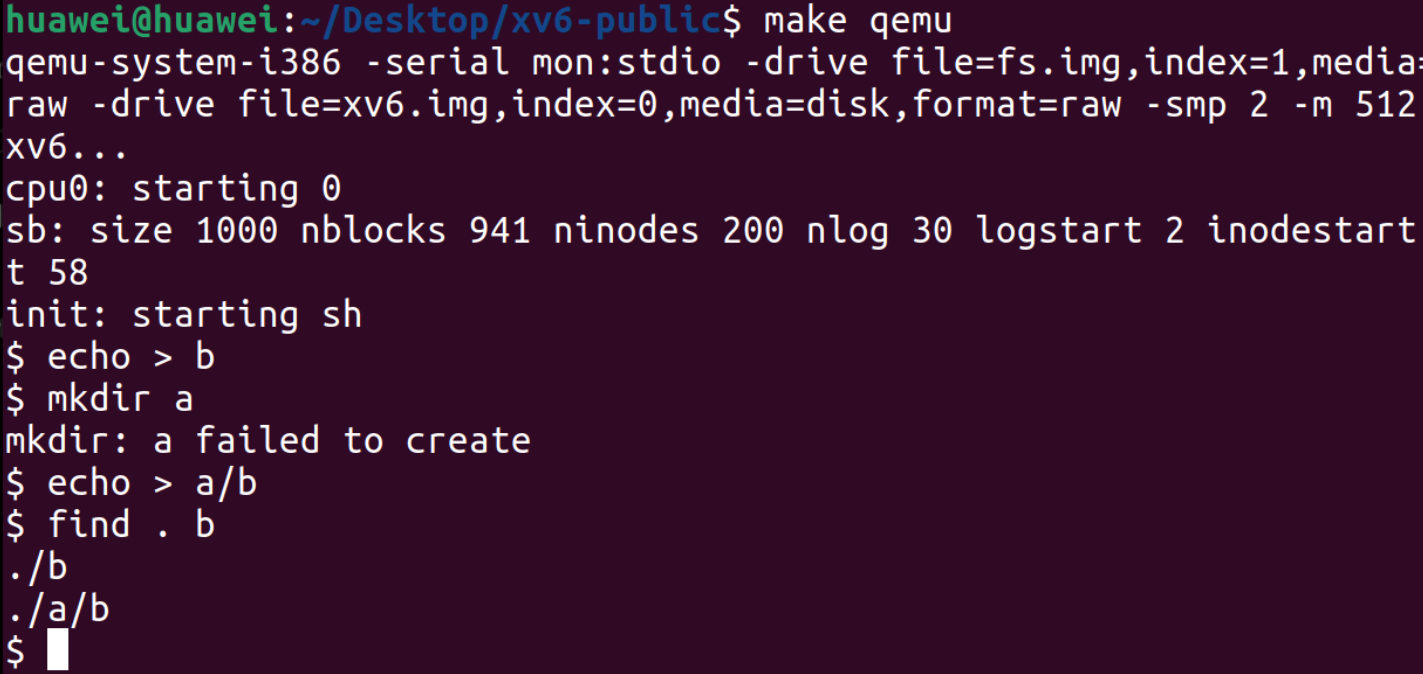


2.编写程序

3.在 Makefile 文件中添加配置：\_find\。



实验结果与心得



之前测试时已创建

在find.c文件中，首先从路径字符串中提取文件名，然后使用 open 系统调用打开指定路径，再使用 fstat 获取文件或目录的元数据信息。如果是文件，检查文件名是否匹配；如果是目录，递归查找该目录下的所有文件和子目录。并对于每个目录，递归调用 find 函数处理其子目录，找到相符文件的就打印。

该实验难度一般，主要要对文件的组织形式和查找过程有较为清晰的认识。

6.xargs

实验目的

编写一个简单的 UNIX xargs 程序，从标准输入中读取行并为每一行运行一个命令，将该行作为命令的参数提供。你的解决方案应该放在 user/xargs.c 中。

实验步骤

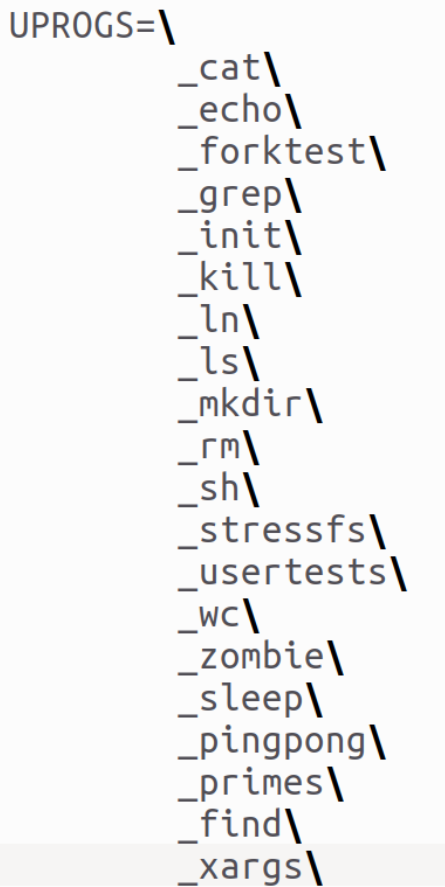
1.创建xargs.c文件



2.编写程序

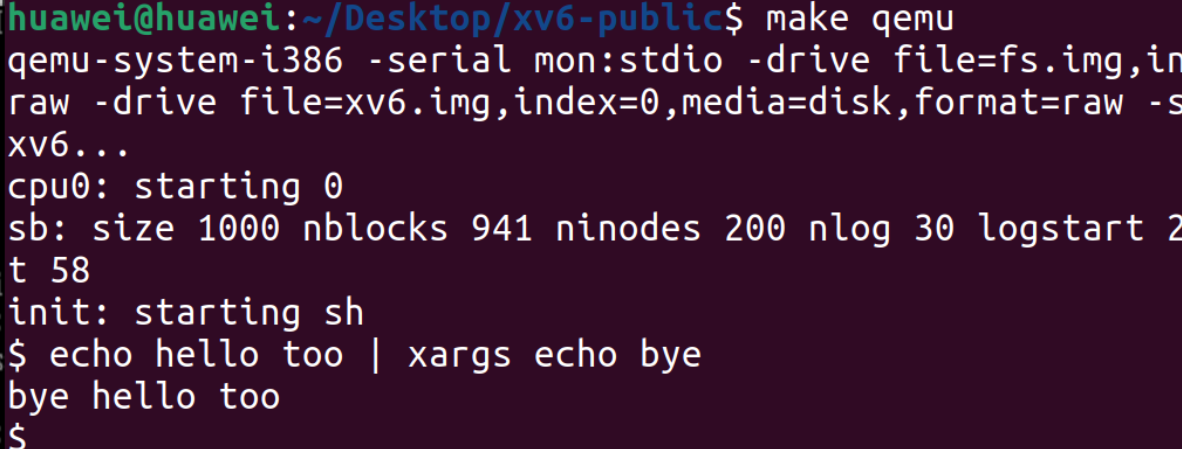


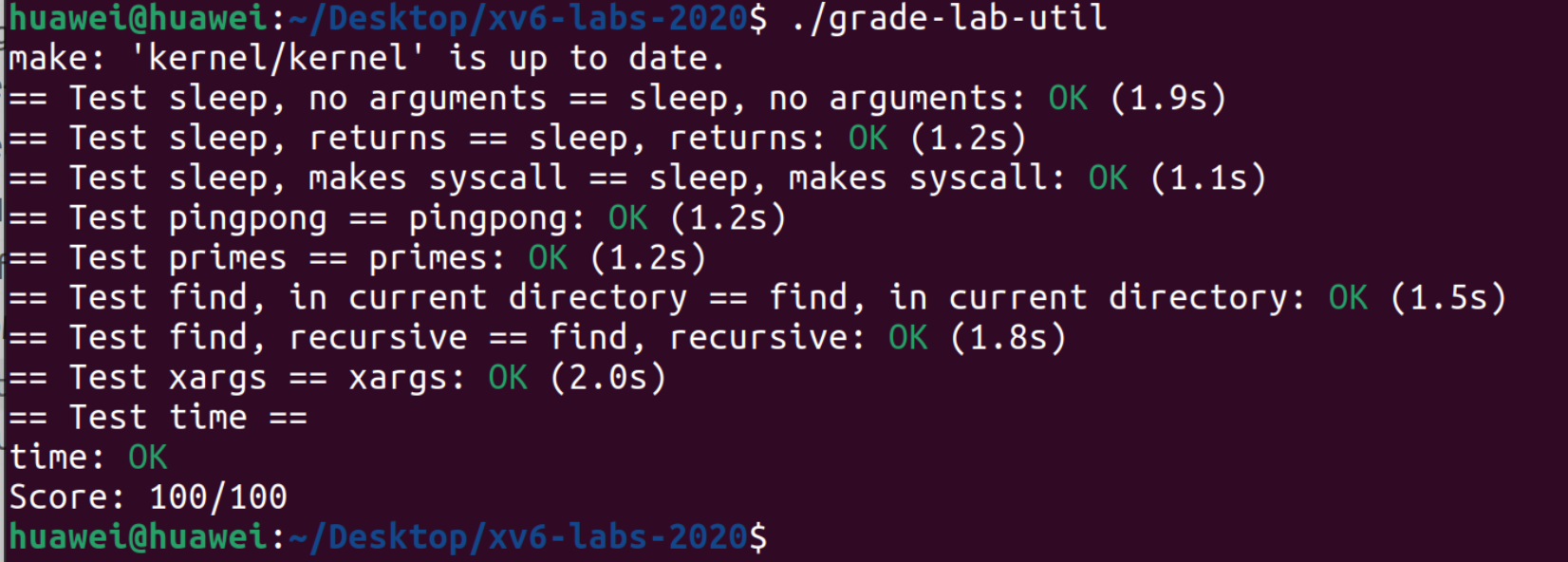
3.在makefile中添加\_xargs\



实验结果与心得

实验结果如下，表明实验成功。





在xargs.c文件中，首先读取标准输入的数据到 buf 缓冲区中且确保以 \0 结尾，并使用 p 指针遍历缓冲区，跳过前导空白字符。解析参数，将参数地址存储到 args 数组中，处理每个参数之间的空白字符。当达到 MAXARGS 限制时，打印错误消息并退出。

创建子进程，并在子进程中使用 exec 执行命令。如果执行失败，打印错误消息并退出。父进程等待子进程完成。重置参数数组，仅保留最初的命令部分。重置缓冲区，以便继续处理新的输入。

实验中，常有参数不匹配的问题，要根据报错信息和文件文本去修正代码，以便可以正确编译。

以上六个实验是对xv6的内核进行的简单实验，通过实验对xv6系统调用有了初步的了解。