# Lab: Xv6 and Unix utilities

|  |  |
| --- | --- |
| 学号： | 2353924 |
| 姓名： | 冯俊财 |
| 课号： | 42028704 |
| 教师： | 王冬青 |

目录

[Lab: Xv6 and Unix utilities 1](#_Toc23224)

[1. Sleep(easy) 1](#_Toc15819)

[1) 实验目的 1](#_Toc7260)

[2) 实验步骤 1](#_Toc4412)

[3) 实验中遇到的问题和解决方法 1](#_Toc18705)

[4) 实验心得 1](#_Toc17833)

[2. pingpong(easy) 2](#_Toc27981)

[1) 实验目的 2](#_Toc16653)

[2) 实验步骤 2](#_Toc15607)

[3) 实验中遇到的问题和解决方法 2](#_Toc7719)

[4) 实验心得 2](#_Toc3202)

[3. primes(moderate)/(hard) 3](#_Toc6559)

[1) 实验目的 3](#_Toc32746)

[2) 实验步骤 3](#_Toc8598)

[3) 实验中遇到的问题和解决方法 3](#_Toc1689)

[4) 实验心得 3](#_Toc25199)

[4. find (moderate) 4](#_Toc13183)

[1) 实验目的 4](#_Toc17163)

[2) 实验步骤 4](#_Toc15074)

[3) 实验中遇到的问题和解决方法 4](#_Toc5082)

[4) 实验心得 4](#_Toc27568)

[5. Xargs(moderate) 6](#_Toc3939)

[1) 实验目的 6](#_Toc26712)

[2) 实验步骤 6](#_Toc60)

[3) 实验中遇到的问题和解决方法 6](#_Toc17270)

[4) 实验心得 6](#_Toc16261)

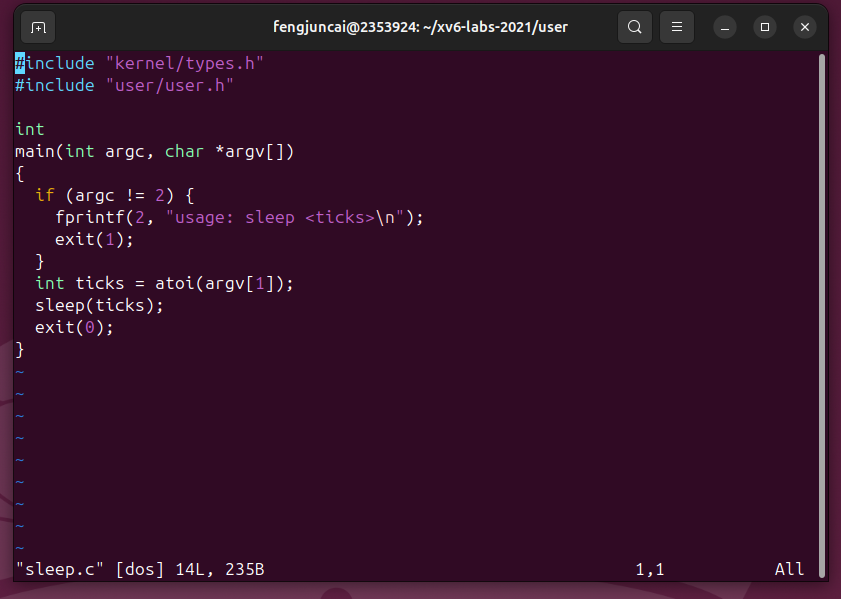
## Sleep(easy)

### 实验目的

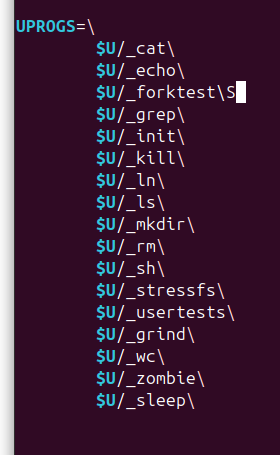
本实验旨在通过实现 sleep 工具，熟悉 Xv6 操作系统的基本系统调用使用方法，理解进程的休眠与唤醒机制，加深对操作系统时钟和进程调度的理解。

### 实验步骤

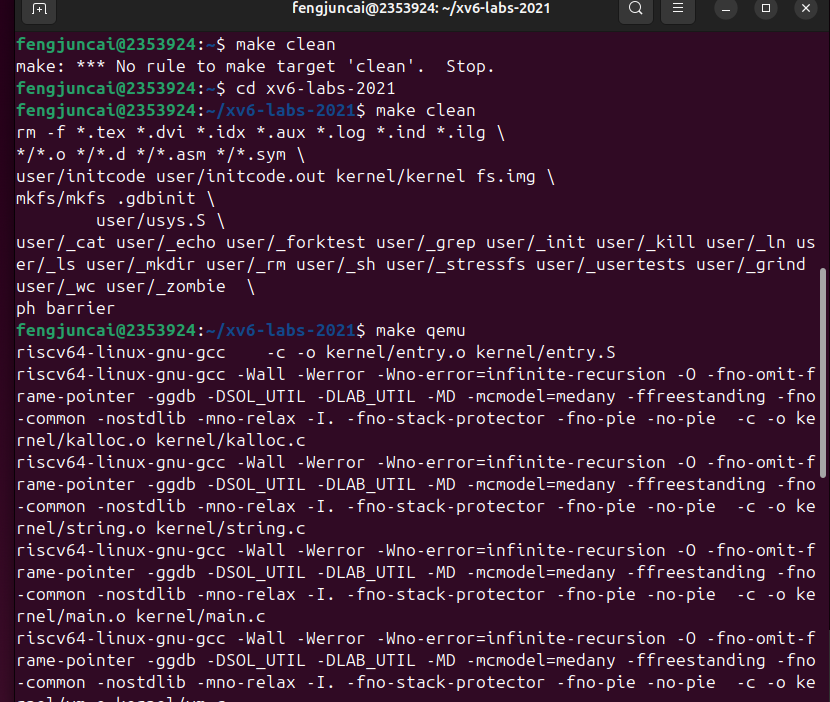
1. 编写代码



1. 更新变量

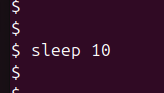


1. 重新编译

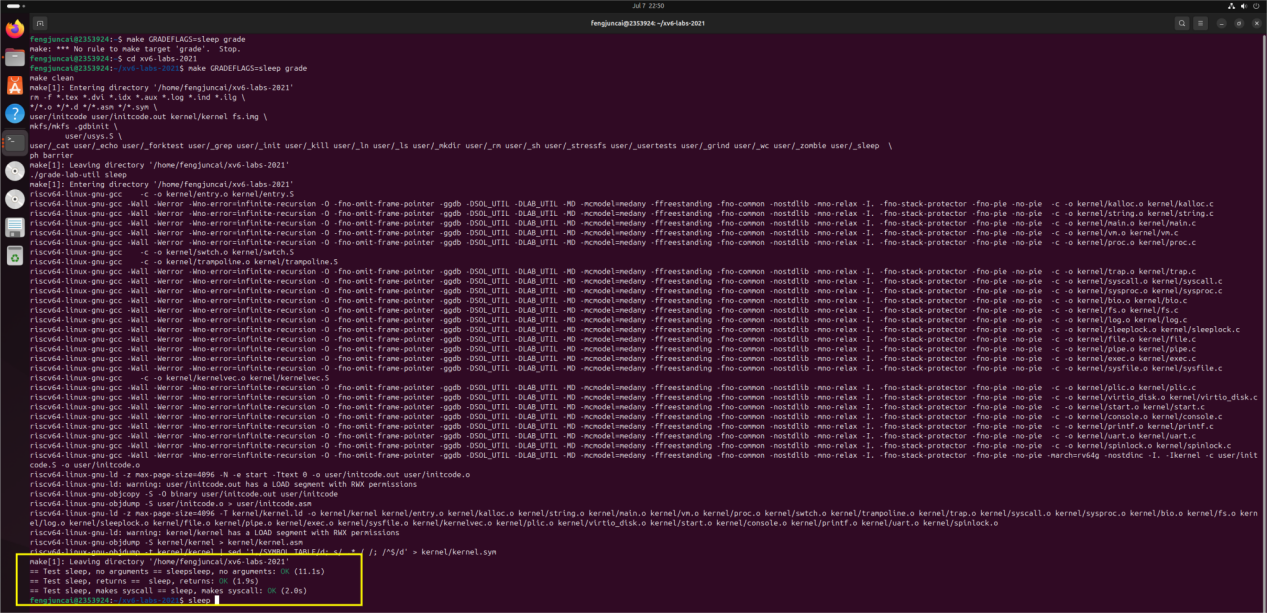


1. 运行代码

|  |
| --- |
| $ cd xv6-labs-2021 # 进入相应的目录  $ make clean #清空  $ make qemu #重新编译  $ sleep 10 # 运行代码 |



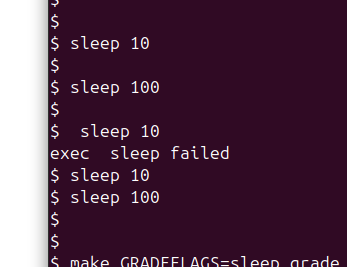
在 xv6 中，sleep 的单位是 “tick”，而不是秒。这里的 ticks 是时钟中断的次数。



运行官方测试程序后，返回OK，通过测试。

### 实验中遇到的问题和解决方法

在运行sleep程序过程中，会突然报错。



可能原因分析：

1. 可执行文件被删除或覆盖：xv6 的文件系统空间有限，用户目录下的文件如果被其他程序或测试覆盖，可能会导致 sleep 文件暂时不存在或损坏。
2. 文件描述符或进程资源耗尽：如果系统中打开的文件太多，或者进程数达到上限，exec 加载新程序时会失败。但你的情况貌似不是持续性错误，后面又能运行，说明资源不是一直被占满。
3. 并发或测试脚本干扰：如果你在运行自动测试（如 make grade）时手动操作 shell，可能会有并发，测试脚本可能会临时删除或替换 sleep，导致你手动执行时找不到。
4. 文件系统不稳定或有bug：xv6 的文件系统是教学用的，偶尔会有未同步的情况，极端情况下可能会有短暂的文件“丢失”。

解决建议：

1. 避免在自动测试运行时手动运行 sleep：测试脚本有可能会清理或重建用户程序，导致你手动运行时偶尔找不到。
2. 重启 xv6 或重新 make clean && make：如果频繁遇到此问题，可以完全重启 xv6 或重新编译、格式化文件系统，保证环境干净。
3. 确认 sleep 是否真的存在：在 shell 下运行 ls 看看 sleep 是否在 / 或 /user 目录下。
4. 减少并发操作：尽量不要在测试脚本运行时频繁手动操作 shell。

### 实验心得

xv6 的 sleep 实现是基于“tick计数”，即每隔一段固定时间（比如10ms或100ms）产生一次时钟中断，sleep(n) 就是让进程挂起n个tick。

由于 tick 是离散的，而不是连续的，所以 sleep 的实际时间会有上下浮动。如果在 tick 快结束时调用 sleep，实际上会等到下一个 tick 才开始计数，所以会“多等”将近一个 tick。

如果在 tick 刚开始时调用 sleep，则误差最小。因此，sleep 的实际耗时通常是 “请求的时间” ± 一个 tick 的误差。

通过本次实验，我理解了xv6中sleep的时间机制基于tick计数而非精确的秒数，由于tick是离散的时钟中断，实际休眠时间会有±1个tick的误差。这让我认识到操作系统时钟管理的复杂性，以及理论与实际实现之间的差异。通过这次实验，我对操作系统的进程调度和时钟中断机制有了更直观的认识。

## pingpong(easy)

### 实验目的

### 实验步骤

### 实验中遇到的问题和解决方法

### 实验心得

## primes(moderate)/(hard)

### 实验目的

### 实验步骤

### 实验中遇到的问题和解决方法

### 实验心得

## find ([moderate](https://pdos.csail.mit.edu/6.828/2021/labs/guidance.html))

### 实验目的

### 实验步骤

### 实验中遇到的问题和解决方法

### 实验心得

## Xargs(moderate)

### 实验目的

### 实验步骤

### 实验中遇到的问题和解决方法

### 实验心得