老师同学们大家好，我做的是上面2020年的这11个实验

这是全部的11个实验

第一个是利用原本的系统调用写一些用户程序，处理linux编程环境

第二个是添加系统调用，告诉我们这些函数怎么陷入内核的，从而区分用户态和内核态

第三个是关于页表，从虚拟地址到物理地址经过三级页表的映射，完成页表的遍历、每个进程添加内核页表

第四个介绍了关于trap机制，让我们修改时钟中断的处理函数，注意进程状态的保存和恢复

第五个是让我们完成请求内存分配的时候只是形式上增加个数字记录，然后用到了再利用page fault去申请内存，完成映射

第六个实验 创建子进程的时候先不拷贝父进程的内存空间，直接共享，然后需要写入的时候才分配子进程的物理空间

第七个实验 是多线程的调度，上下文切换和同步的实现

第八个实验 给每个CPU的freelist空闲快链表上锁保证分配正常

第九个实验 是弄了二级索引扩充单个文件的大小，然后是实现软链接快捷方式

第十个实验 利用虚拟内存区域这个结构体记录相关信息，然后不同进程可以映射到同一个磁盘上的文件从而进行共享，然后就可以直接访问而不用调用read()、write()这些系统调用，第十一个实验 是实现网卡驱动的发送接收功能

接下来主要展示一下几个方面遇到的问题

首先是**环境配置**上，首先要安装Linux系统，本来是之前在virtualBox上安装了Ubuntu22.04版本的虚拟机了（图），想着直接拿来用，但是第一个问题就是它的qemu他版本不兼容，导致make qemu这个命令执行后一直卡在某个地方，也不能键盘命令结束进程，只能关掉终端然后再进去，但是问题还没解决（图）

最后是在看到它是版本不对，而且官网上也说建议安装Ubuntu20.04版本的，我这边因为安装虚拟机太久了，就换了另一种方式，也是询问其他同学才直到的，用WSL直接在微软商店里面下载一个小小的程序和一些配置就可以运行（图）

然后接着安装官网上的命令安装即可

还有一问题就是WSL自带的那个终端编辑起来较为不方便，主要是不能用鼠标点，只能用vim相关指令，记得也不是很熟

解决方法也是问了同学以及上网查询资料，直接用电脑里面的vsCode远程连接这个WSL，这样不仅可以较方便地编辑文件，可以直接在里面启动终端执行命令

然后是比较困扰的问题和印象比较深的实验心得

首先是，这个构建系统调用的**实验二**，这个应该是之后很多实验的基础，带我们从用户态陷入到内核态然后执行相关命令和功能，以下是相关步骤：

在用户程序中也就是user目录下添加的程序中调用系统调用

在user/user.h添加系统调用的函数原型声明

系统调用序号的宏定义在kernel/syscall.h文件中

在user/usys.pl文件中添加入口函数，这是perl脚本，可以生成usys.S文件

usys.S文件中设置了a7寄存器存储系统调用号，ecall指令进行系统调用

然后转到kernel/syscall.c文件中的syscall()函数通过添加的系统调用号找到kernel/sysproc.c文件中的对应添加的函数并执行

同时，我在做实验的时候还有一个问题，这些系统调用都没有显式的传递参数的方法，那么参数是怎么获得的呢  
我们可以看到在kernel/sysproc.c文件中对系统调用的实现，会使用argint()函数获得参数，它可以读取在a0-a5的寄存器中传递的系统调用的参数

然后是**lab3关于页表的实验**，做之前上课讲的理论还可以理解，但是实际到这个实验做的非常头疼，特别是这个实验的第二个和第三个实验做了三四遍，里面有很多小细节一不小心就是各种panic的错误，通过这个真的可以学到很多东西

首先第一个小实验叫我们实现打印页表项和对应的物理地址，这个应该是后两个的基础，有点要注意的就是对于pagetable\_t和pte\_t和uint64单纯的物理地址的转换

然后做实验的时候我们可以看到PTE2PA(pte)从页表项到物理地址的转换是右移10位再左移12位，那时候跟舍友讨论了很久，之后再结合书上这张图才弄明白。虚拟地址存的是三个页表中的偏移量和物理地址的偏移量，然后第一次通过satp寄存器找到一级页表首地址，通过偏移量找到其中的页表项，从而又得到二级页表的首地址。

将低10位的标志位去掉，然后左移12位用0填充作为offset

然后第二个实验是叫我们给每个进程都来个内核页表，我这里遇到的问题是panic：kerneltrap原因是procalloc的时候忘记增加各自进程的内页页表的初始化；还有问题是usertests的时候卡在copyin那个测试项，原因是scheduler中没有切换回全局的内核页表

还有一个遇到过几次的问题，就是添加自旋锁和proc头文件的定义，想要利用proc的成员函数，但开始报错没有proc结构体的定义，于是我们添加proc.h头文件，然后又报错没有sinlock的定义，然后才编译成功

### 第四个实验trap

关于中断机制，这里完成了用户态到内核态的陷入

这里主要问题一个是第三个实验遇到了他说alarm handler多次调用，之后这个qemu给中止了，解决方法是给他弄一个标志位记录，解决方法是比较简单但是这个问题的出现是之前没有想到的，为什么会多次陷入呢

### 第五个实验lazy page allocation

分配内存的时候只是形式上增加个数字记录，然后用到了再利用page fault去申请内存，完成映射

主要就是添加自旋锁和proc头文件的定义，想要利用proc的成员函数，但开始报错没有proc结构体的定义，于是我们添加proc.h头文件，然后又报错没有sinlock的定义，然后才编译成功

### 第六个实验copy-on-write fork

创建子进程的时候先不拷贝父进程的内存空间，直接共享，然后需要写入的时候才分配子进程的物理空间

walkcowaddr()函数的错误判断没有直接跳到end处，而是继续执行下去，就出现了问题  
在之前的学习中，确实很少使用这种语法，因此有些许不熟悉，不过此次将能得到很好的锻炼

### 第七个实验multithreading

这个是多线程的调度，上下文切换和同步的实现

显示类型转换

thread\_switch((uint64)&t->thrdctx, (uint64)&current\_thread->thrdctx);

还有第二个小实验是代码改了和没改一样，之后上网查了才知道是要clean以下然后重新编译

### 第八个实验lock

给每个CPU的freelist空闲快链表上锁保证分配正常

这个问题完全是基础问题，数据结构那块，对链表差点忘了，弄了好久没弄成功，这个在纸上话清楚会比较好

同时这个也是要make clean以下然后再重新编译，不然会有panic: freeing free block的错误

### 第九个实验file system

首先是弄了二级索引扩充单个文件的大小，然后是实现软链接快捷方式

修改 sys\_open() 函数错误导致一直无法启动xv6系统，因此我们多找几个网上的参考

sys\_open() 函数中原来的 ilock(ip) 忘记删除，导致多次 ilock() 函数的调用，删掉就好了

### 第十个实验memory map

利用虚拟内存区域这个结构体记录位置、容量、权限一些信息，然后不同进程可以映射到同一个磁盘上的文件从而进行共享，然后就可以直接访问而不用调用read()、write()这些系统调用，他们是什么呢，他们是把数据从磁盘拷到内存然后再拷到用户进程里面，比我们这个少了一次拷贝的次数

### 第十一个实验net working

这个是实现网卡驱动的发送接收功能

这个实验难点还是在于理解网卡的这些概念，它通过以太网这个局域网技术来实现发送和接收数据