实验七 分组实验

谷建华

2022-12-3 v0.2

此次实验为大作业，大家以两人一组，选取一个实验题目实现。

1. 编写线程相关系统调用：
    - 线程系统调用需要实现的如下：
    - pthread_create 线程创建
    - pthread_exit 线程退出
    - pthread_join 线程等待
    - 按照glibc库给出的语义实现
2. 实现写时复制：
    - 通过修改fork系统调用，使系统调用并不是完全复制父进程的所有内存信息，而是与父进程内存共享，但是所有物理页修改成只读权限不允许修改。
    - 如果进程要执行写操作，就会触发Page Fault，对处理函数修改使得写时复制能够实现。
    - 你实现的写时复制需要考虑进程退出的情况，对页面内存的回收是否合适。
3. 实现CFS调度策略+实现增量式sleep系统调用实现：
    - CFS是linux中常用的调度策略，一个大家比较熟悉的命令nice就与之有关，通过进程的nice值判断如何分配多少时间片。
    - 编写nice系统调用控制不同进程的分配时间片权重。
    - 增量式sleep上课讲过，通过维护sleep队列中的等待时间增量实现O(1)弹出，插入为O(n)的sleep实现
4. 实现缺页中断+页面换进换出：
    - 在之前的exec实现中都是直接将ELF文件整个复制进来，再根据ELF文件信息将数据搬迁到用户指定位置。但是如果ELF文件很大会导致加载非常的慢。
    - 缺页中断的思想就是不先加载完整的ELF，而是先让用户执行，当用户执行到一个不存在的页会触发Page Fault，然后在处理程序中对相应是数据段进行加载。
    - 实现了缺页的同时也要实现换页，换页实际上是缓解内存不足的一种做法，它将进程的页面信息存放到可持久化存储处，并将页面内存释放出

来给其他进程用。

5. 锁操作的实现(自旋锁)，实现P、V信号量操作：
    - 实现`pthread_spin_*`，`pthread_mutex_*`等线程相关同步互斥原语。
    - 按照glibc库给出的语义实现。
6. 通过消息队列实现简单的进程间通信：
    - 进程间通信有很多种方式，socket通讯是其中的一种，但是学生实现不出来socket这些系统调用，所以简化问题为使用消息队列通信。
    - 这个可以参考的地方比较少，更多体现实现的创造性，没有标准答案。
7. 图形界面GUI实现：
    - 目前OS都是命令行界面，但是可以考虑探索一下图形界面的可能性。
    - 实现图形界面有部分开源程序miniGUI、openGUI。
    - 这是涉及到framebuffer的知识，如果学生能够采用AMD技术或AGP或PCIe来实现高效的图形处理，将会获得更好的成绩。
8. 实现ext2文件系统：
    - 虽然现在linux最常使用的底层文件系统为ext4，但是ext2文件系统成为一代经典。
    - 可以先考虑简单实现出`read`,`write`,`open`,`close`,`create`这些vfs接口。
9. 实现qemu虚拟机网络通讯：
    - 探索在qemu虚拟机上与外界网络通讯的可能性，只需要实现mac层通讯，就比如外界向虚拟机对应的端口发送信息，虚拟机能够接受到对应的数据并做出应答。
10. 实现execve系统调用：
    - 之前我们实现的都是exec，但是实际上这个系统调用还有`v,e`两个参数，其中`v`用于向程序提供参数，`e`用于向程序提供环境变量。
11. 实现mount/umount系统调用：
    - 实现mount/umount系统调用实现将文件挂载到指定目录下，并能够实现挂载目录中文件的创建和删除
    - 建议先实现系统调用mount和unmount，功能和参数形式参考Linux。然后实现一个应用程序mount和unmount，这两个程序可以在命令行下运行。
12. 实现signal信号机制：
    - 实现signal需要的系统调用如下：
    - kill 向指定进程发送信号
    - sigaction 用户可以根据信号的类型提供相应的用户态处理函数

    - sigreturn 用户在执行完用户态处理函数后通过该系统调用告知内核处理结束
13. 实现orange树型目录结构的实现：
    - 目前给的实验代码中的用的底层文件系统是orange作者自创的文件系统，它是一个扁平化文件系统，就意味着没有文件夹这种结构
    - 这种结构可以说非常的离谱，所以需要考虑将其改造变为树状结构，让其有创建文件夹的功能，这个是你可以自由发挥的地方。
14. 块设备（硬盘）缓冲区管理：
    - 目前给的实验代码中它对缓冲区的管理可以说是几乎没有，遇到一个请求就开始读/写，这样有一个不好的地方就是非常的磁盘I/O，相比于从块设备中读入，在内存中修改明显更快。
    - 所以就需要在磁盘中构建一个缓冲区用于临时存放磁盘中的数据拷贝，所有的读写操作都在内存中完成，当缓冲区满时才将部分数据写回磁盘。
    - 建议每个缓冲区以文件系统的文件块（一般是4K）为单位，而不是以扇区为单位实施管理。
15. 实现buddy分配内存算法：
    - buddy伙伴算法可以说是linux中非常经典的分配内存手段，它以页(4kb)为单位，分配连续2的幂次个页，现在你需要实现该分配算法。
16. 实现slab分配内存算法：
    - slab分配算法也是非常经典的分配内存手段，相比于buddy伙伴算法这种分配大内存的（因为以页为单位），slab更擅长分配小内存，现在你需要实现该分配算法。
17. 重写FAT32文件系统：
    - 目前给的实验代码中的FAT32不知道怎么开发的，就实现的非常的难看，而且很多功能都没有实现好，需要对其重写。
18. 实现一个基于内存的文件系统：
    - 在内核中开辟一段内存用于存放文件系统数据（文件系统格式任意）
    - 这是一个临时的文件系统，它并不能可持久化存储，当虚拟机资源释放时文件系统也被销毁
    - 通过vfs的API能够对该文件系统进行访问
19. 转义序列支持的完善+对tty的支持：
    - 这轮教学中加入了kprintf和CSI转义序列扩展了输出字符的方式，但是支持比较简陋，不支持更多转义序列和对终端支持差（不能支持终端屏幕滚动）

    - 你需要结合实验现有代码对其改造，能够适配CSI转义序列和kprintf，摒弃原先的`disp_*`输出方式，这种输出方式可以说是非常难受了。
    - 你的实现可以用于下一轮试点班教学，能够实现更多有意思的东西。
20. 实现poll或select系统调用：
    - 这是一个检测文件描述符变化的系统调用，它会在内核态中轮询，检测文件描述符变化，如果有相应的变化会通知用户态执行。
    - 实现该系统调用后可以修改shell程序，使得获取输入不再在用户态中轮询。
21. 实现pipe,dup系统调用：
    - 管道是unix中非常标志性的一个特性，也是进程间通讯的一种方式
    - 通过pipe创建管道文件描述符，再通过close，dup系统调用实现文件描述符的移动。
    - 实现该系统调用后可以修改shell程序，使得通过管道实现多个进程间数据的传输。
22. loader的C语言替换：
    - 这轮教学中由于对大kernel的需要将loader中实现从汇编转变为C语言，读取kernel也是在保护模式下进行。
    - 你需要尽可能将loader中的汇编语义翻译成C语言，保护模式加载kernel（要求高鲁棒性），获取内存信息，和分页。
    - 尽可能采用C语言来实现原来主要由汇编实现的loader程序，一定要注意尽量保持loader与miniOS其他部分的接口保持不变。
    - 你的实现可以用于下一轮试点班教学，能够实现更多有意思的东西。
23. 探索启动64位x86系统的可能性：
    - 探索一下如何qemu启用64位系统，而不是32位i386。
    - 不需要把整个miniOS都改造成支持64位的程序。初步建议是，miniOS的boot和loader部分都保持不变。进入kernel后对64位的相关寄存器，例如GDT，LDT，IDT，页表等重新进行初始化
24. 移植多核版本miniOS-MP
    - miniOS-MP是基于miniOS-v1.2.7构建的多核操作系统，目前已经稳定运行在多核qemu上。
    - 本实验希望将miniOS-MP移植到miniOS-v1.3.2中，使miniOS-v1.3.2可以运行在多核qemu上。
    - 任务1：移植miniOS-MP的多核启动程序到miniOS-v1.3.2中，使miniOS-v1.3.2实现多核硬盘启动。
    - 任务2：在任务1的基础上，使多核的miniOS-v1.3.2可以响应多核架构

下的硬盘中断。

- miniOS-MP是刘明轩的本科毕业设计，欢迎感兴趣的同学联系刘明轩询问细节。