功能实现总结

- 重写了系统调用 `sys_get_time` 和 `sys_task_info` 以支持虚拟内存管理
- 实现了 `mmap` 系统调用，支持将物理内存映射到指定的虚拟地址空间，并设置相应的读/写/执行权限
- 实现了 `munmap` 系统调用，用于解除指定虚拟地址区间的内存映射
- 添加了虚拟地址范围检查和内存页映射状态的管理功能

页表与缺页机制

1. SV39页表项组成

- **PPN[2:0]**: 物理页号，共44位
- **RSW**: 预留给S态软件使用，2位
- **D**: Dirty位，表示页面是否被修改
- **A**: Access位，表示页面是否被访问
- **G**: Global位，表示是否在所有地址空间有效
- **U**: User位，控制用户态访问权限
- **X,W,R**: 执行/写/读权限位
- **V**: Valid位，表示页表项是否有效

2. 缺页相关

缺页可能导致的异常

- Load/Store page fault
- Instruction page fault

重要寄存器

- stval: 记录发生异常的虚拟地址
- sepc: 保存触发异常指令的PC
- scause: 指示异常原因

Lazy策略优势

- 节省内存空间
- 减少启动时间

- 提高内存利用率

SV39页表大小估算

- 10G = 2^33，需要2^21个页表项
- 三级页表总大小约为几十MB级别

Lazy实现思路

1. mmap只建立虚拟地址映射
2. 访问时触发缺页
3. 缺页处理程序分配物理页并建立映射

Swap页表项表现

- V位置0
- 其他位用于存储磁盘位置信息

3. 双页表与单页表

单页表更换

- 仅需修改satp寄存器

控制用户访问

- 通过页表项的U位控制

单页表优势

- TLB效率更高
- 内存开销小
- 切换开销小

页表切换时机

- 双页表：进入/退出内核态时
- 单页表：进程切换时

> 我从未使用过他人的代码，不管是原封不动地复制，还是经过了某些等价转换。 我未曾也不会向他人（含此后各届同学）复制或公开我的实验代码，我有义务妥善保管好它们。 我提交至本实验的评测系统的代码，均无意于破坏或妨碍任何计算机系统的正常运转。 我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按“-100”分计。