实验报告

系统调用

声明函数

- 先在syscall.asm中设置不同系统调用对应的数值，在global.c中的sys_call_table中添加与数值对应的处理函数
- 通过eax的值判断调用哪个函数

传递参数

- c语言利用栈传递参数，从右向左压栈
- 利用寄存器传递参数，压入内核栈。再从内核栈中调出参数传给c语言的函数

milli_seconds实现（代码中命名为sleep）

- 为进程新增wake_time变量，设置进程未来醒来的时间为 get_ticks()+milli_seconds/(1000/HZ)
- 改写schedule函数，比较get_ticks()与进程的wake_time变量的大小关系
- 当大于wake_time时表明进程可以醒来

读者写者问题

信号量实现

- 定义semaphore结构体，包含value与process_list
- value用于保存信号值，process_list用于保存等待进程，等待唤醒
- 在进程进入process_list等待队列的同时，设置进程的wake_for_sem为true，用于schedule调度。

读者优先/写者优先

- 读者优先于写者优先参考ppt伪码，利用实现的P、V系统调用完成
- 为每个进程新增wait_for_sem变量，用于判断是否在等待信号量
- 在schedule中增加判断条件
    - 如果wait_for_sem为true，表明在等待信号量，不能调度该进程
    - 如果wait_for_sem为false，表用不用等待信号量，可以调度该进程

饥饿问题

- 读者优先/写者优先都可能会导致写者或读者饿死
- 可以在读/写者优先时，让reader/writer完成操作后睡一段时间，可以在一定程度缓解饿死问题

打印函数的定时调用

- 利用自己实现的sleep，让打印函数定时睡眠
- 打印函数不存在pv操作，没有饿死现象，定时调用sleep即可