实验2

任务1 进程优先级

- 实现ps命令（ps.c）
  - 命令格式如下:

PID	PPID	PRI	MEM	STATE	CMD
1	N/A	10	12288	SLEEPING	init
2	1	10	16384	SLEEPING	sh
4	2	10	12288	RUNNING	ps

  - 在proc.c中实现getptable()系统调用
  - 在ps命令中调用getptable()实现进程列表

任务1 进程优先级

- 实现一个用户空间程序（nice.c），用于在运行时修改进程优先级。nice命令的调用方式如下
  - nice pid priority （[1, 20]中，数值越小，优先级越高）
  - 实现系统调用（即setpriority(pid, priority)）。

任务2 信号量实现

实现下列函数(proc.c)：

int sem_init(int sem, int value)

初始化信号量结构体的值。

int sem：该参数是信号量的编号；

int value：该参数是信号量的值。

如果初始化成功，返回值为 0，否则返回 - 1。

int sem_destroy(int sem)

销毁信号量结构体。

int sem：该参数是信号量的编号。

返回值为 0。

任务2 信号量实现

int sem_wait(int sem, int count)

允许进程 / 线程通过使用信号量来获得公共资源的独占权，或者等待其他进程释放该资源。

int sem：该参数是信号量的编号；

int count：该参数是用于与信号量的值进行比较的数值。

返回值为 0。

int sem_signal(int sem, int count)

允许进程释放通过信号量独占的资源。

int sem：该参数是信号量的编号；

int count：该参数是用于与信号量的值进行比较的数值。

返回值为 0。

任务3 使用信号量解决文件读写互斥

- 创建文件sem_test.c，实现下列过程：
  - 创建10个进程，每个进程从文件读取数值并加1，该过程重复50次
  - 保证通过互斥后，最终写入文件的数值为全体进程计数的和，即500。