实验四细节补充

1. 进程调度：

```c
/* proc.c */
PUBLIC void schedule()
{
    PROCESS* p; // 修改此结构体
    int current_tick = get_ticks();

    while (1) {
        p_proc_ready++;
        if (p_proc_ready >= proc_table + NR_TASKS) {
                p_proc_ready = proc_table;
        }
        if (p_proc_ready->waiting_semophore == 0 &&
            p_proc_ready->wake_tick <= current_tick) {
                break; // 寻找到进程
        }
    }
}
```

2. 进程睡眠的实现方式

```c
/* clock.c */
PUBLIC void milli_delay(int milli_sec)
{
        int t = get_ticks();
        while(((get_ticks() - t) * 1000 / HZ) < milli_sec) {}
}
/* klib.c */
PUBLIC void delay(int time)
{
    int i, j, k;
    for(k=0;k<time;k++){
        for(i=0;i<10;i++){
            for(j=0;j<10000;j++){}
        }
    }
}
```

建议读者进程使用milli_delay，可以模拟多个读者并行(同时)读文件的场景。输出进程采用系统调用。

3. 时间片公式：

$$time * 1000 / HZ$$

```
/* proc.c */
PUBLIC void process_sleep(int milliseconds) {
    p_proc_ready->wake = get_ticks() + (milliseconds / (1000 / HZ));
    schedule();
}
```

4. 解决饿死问题建议方案
    a. 公平读写
    b. 优先级调度