

实验四细节补充

1. 进程调度:

```
1  /* proc.c */
2  PUBLIC void schedule()
3  {
4      PROCESS* p; // 修改此结构体
5      int current_tick = get_ticks();
6
7      while (1) {
8          p_proc_ready++;
9          if (p_proc_ready >= proc_table + NR_TASKS) {
10             p_proc_ready = proc_table;
11         }
12         if (p_proc_ready->waiting_semaphore == 0 &&
13             p_proc_ready->wake_tick <= current_tick) {
14             break; // 寻找到进程
15         }
16     }
17 }
```

2. 进程睡眠的实现方式

```
1  /* clock.c */
2  PUBLIC void milli_delay(int milli_sec)
3  {
4      int t = get_ticks();
5      while(((get_ticks() - t) * 1000 / HZ) < milli_sec) {}
6  }
7  /* klib.c */
8  PUBLIC void delay(int time)
9  {
10     int i, j, k;
11     for(k=0;k<time;k++){
12         for(i=0;i<10;i++){
13             for(j=0;j<10000;j++){
14             }
15         }
16     }
```

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：

$$time * 1000 / HZ$$

```
1 /* proc.c */
2 PUBLIC void process_sleep(int milliseconds) {
3     p_proc_ready->wake = get_ticks() + (milliseconds / (1000 / HZ));
4     schedule();
5 }
```

4. 解决饿死问题建议方案

- a. 公平读写
- b. 优先级调度