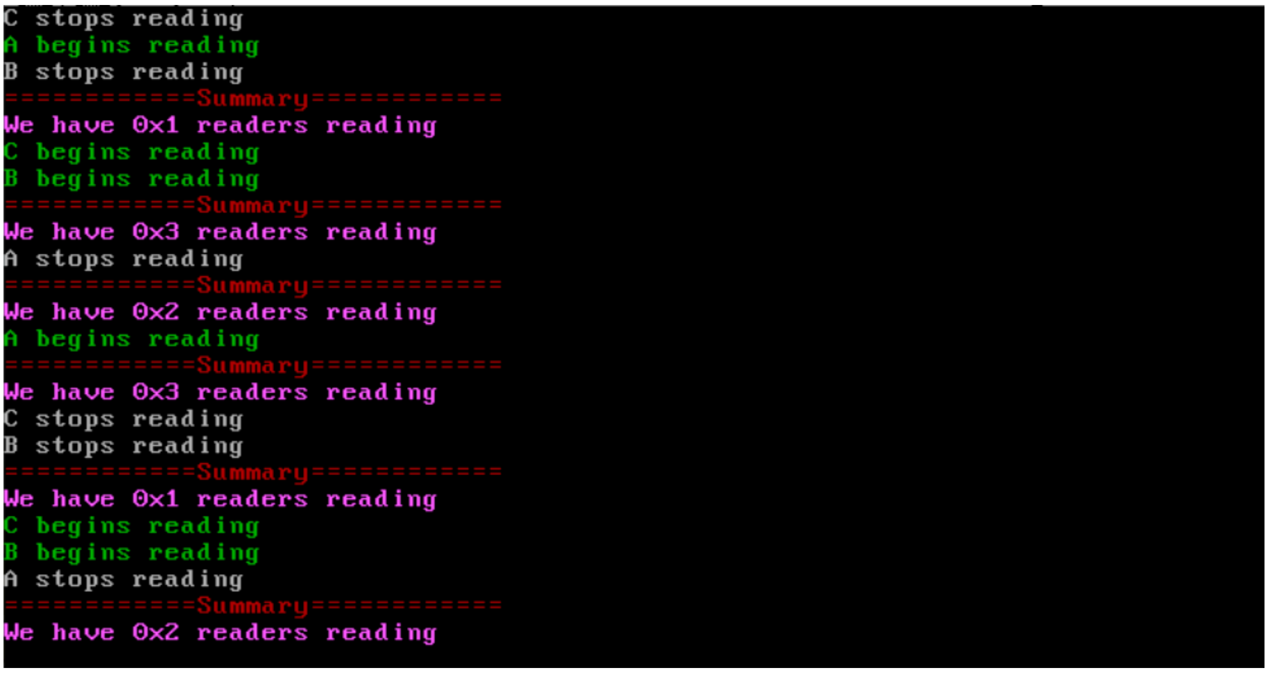
Lab4 实验报告

陆放明 171250660

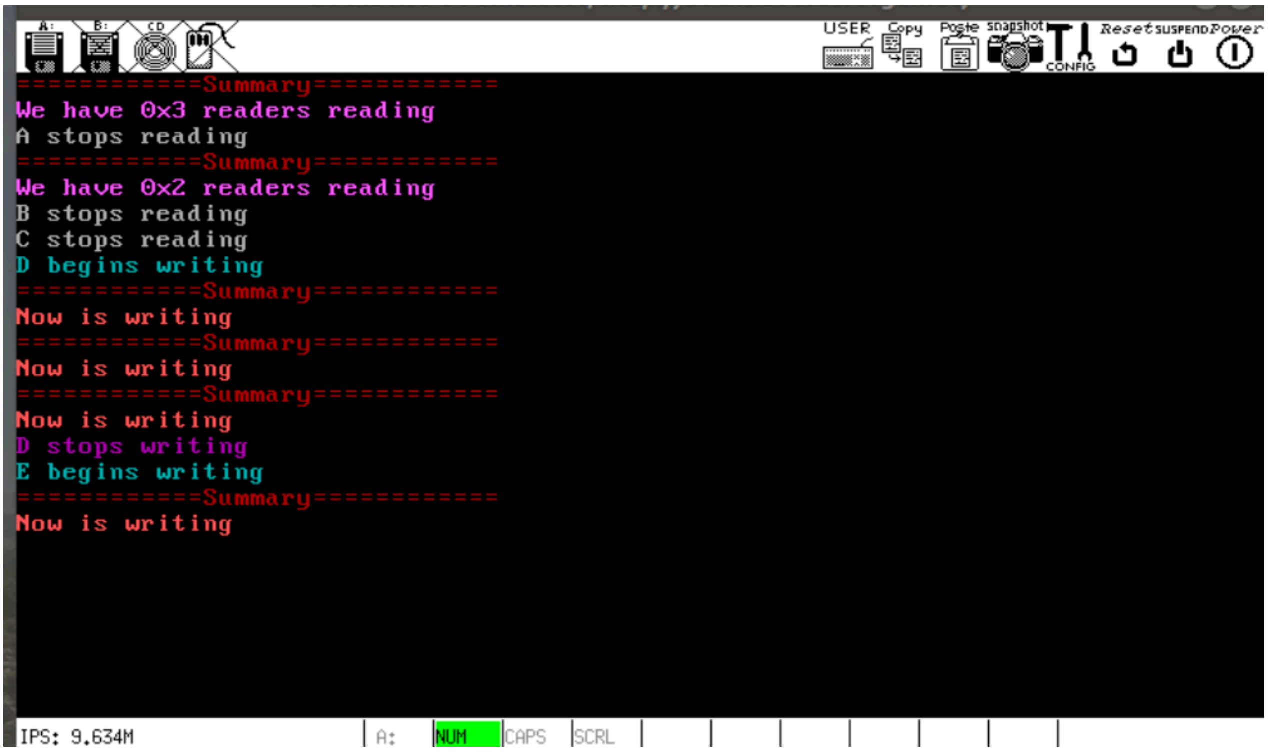
1. 运行截图

1.1 读者优先



```
C stops reading
A begins reading
B stops reading
============Summary============
We have 0x1 readers reading
C begins reading
B begins reading
============Summary============
We have 0x3 readers reading
A stops reading
============Summary============
We have 0x2 readers reading
A begins reading
============Summary============
We have 0x3 readers reading
C stops reading
B stops reading
============Summary============
We have 0x1 readers reading
C begins reading
B begins reading
A stops reading
============Summary============
We have 0x2 readers reading
```

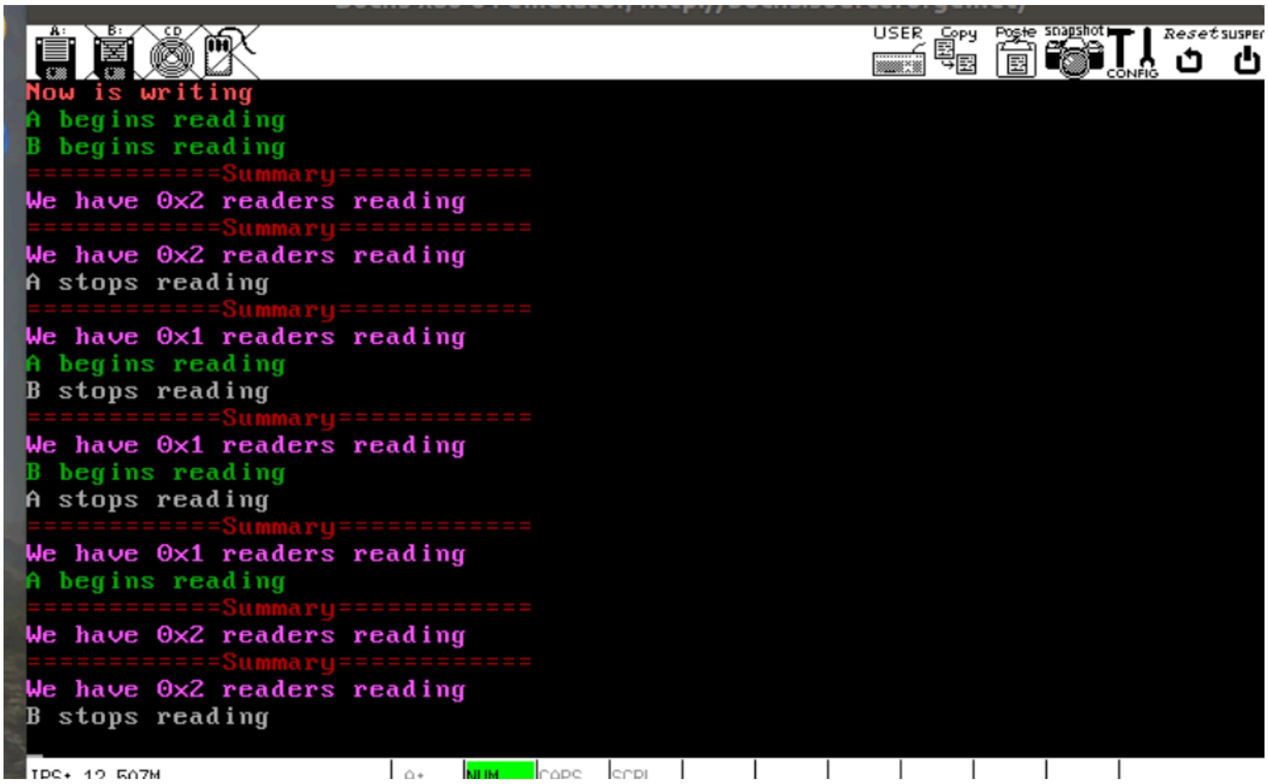
1.2 写者优先



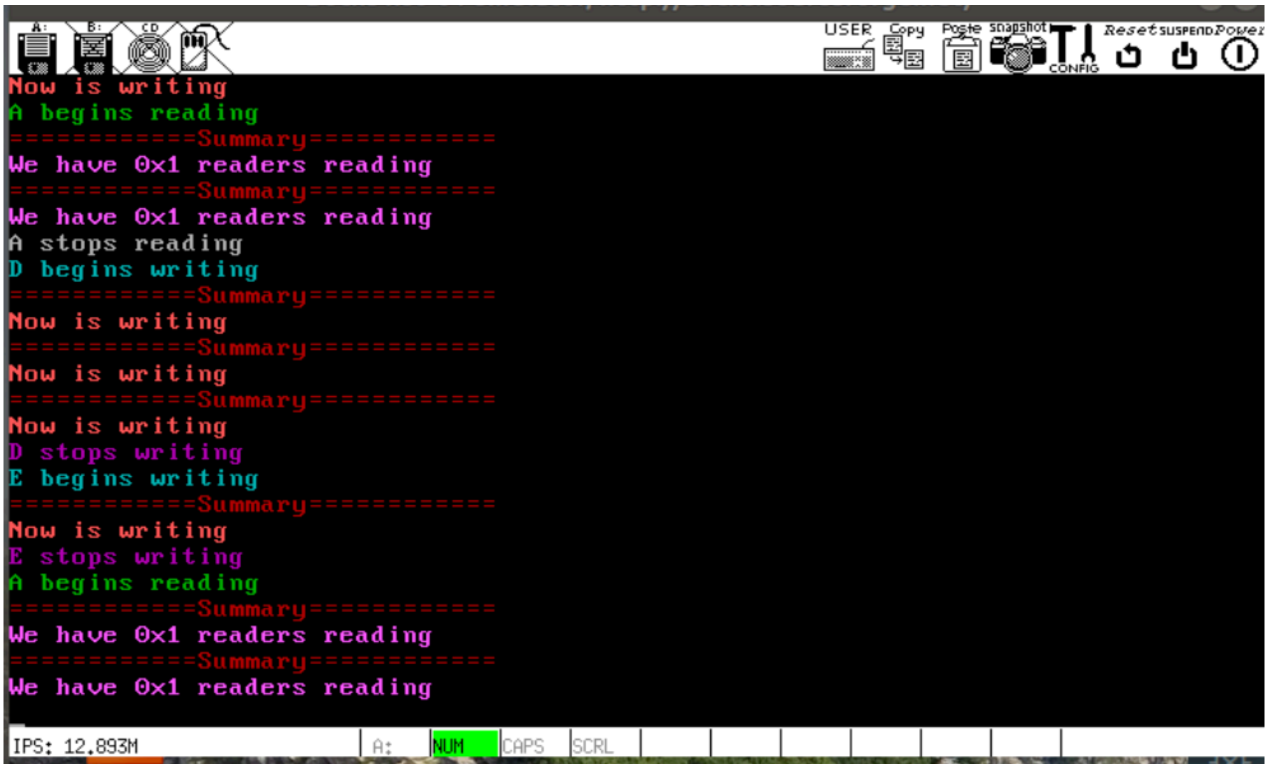
```
============Summary============
We have 0x3 readers reading
A stops reading
============Summary============
We have 0x2 readers reading
B stops reading
C stops reading
D begins writing
============Summary============
Now is writing
============Summary============
Now is writing
============Summary============
Now is writing
D stops writing
E begins writing
============Summary============
Now is writing
```

```
IPS: 9.634M    A:    NUM    CAPS    SCRL
```

1.3 至多2个读者同时读

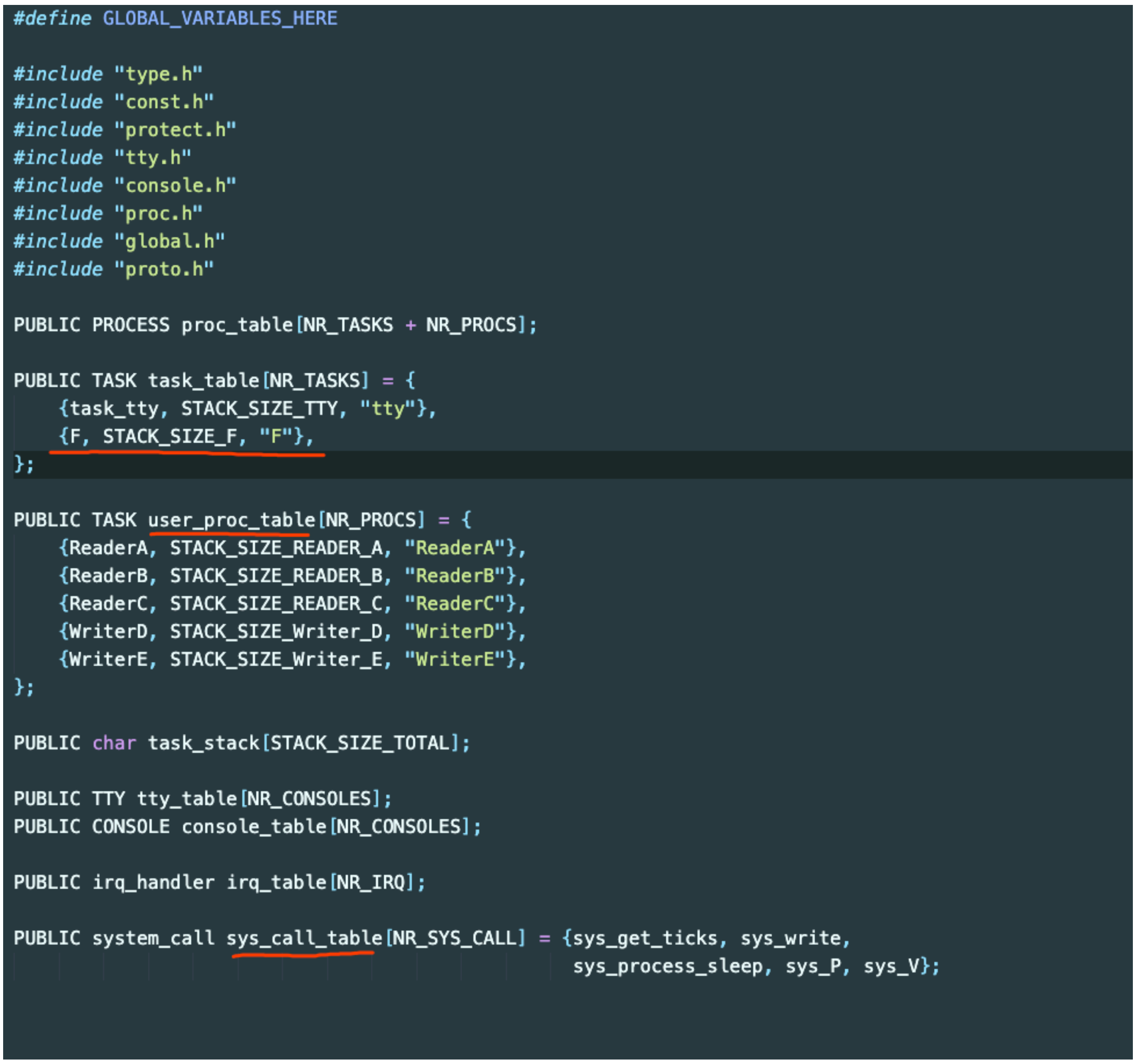

```
Now is writing
A begins reading
B begins reading
===========Summary===========
We have 0x2 readers reading
===========Summary===========
We have 0x2 readers reading
A stops reading
===========Summary===========
We have 0x1 readers reading
A begins reading
B stops reading
===========Summary===========
We have 0x1 readers reading
B begins reading
A stops reading
===========Summary===========
We have 0x1 readers reading
A begins reading
===========Summary===========
We have 0x2 readers reading
===========Summary===========
We have 0x2 readers reading
B stops reading
```

1.4 至多1个读者读


```
Now is writing
A begins reading
===========Summary===========
We have 0x1 readers reading
===========Summary===========
We have 0x1 readers reading
A stops reading
D begins writing
===========Summary===========
Now is writing
===========Summary===========
Now is writing
===========Summary===========
Now is writing
D stops writing
E begins writing
===========Summary===========
Now is writing
E stops writing
A begins reading
===========Summary===========
We have 0x1 readers reading
===========Summary===========
We have 0x1 readers reading
```
IPS: 12.893M

2. 修改位置

2.1 进程和系统调用的添加



```
#define GLOBAL_VARIABLES_HERE

#include "type.h"
#include "const.h"
#include "protect.h"
#include "tty.h"
#include "console.h"
#include "proc.h"
#include "global.h"
#include "proto.h"

PUBLIC PROCESS proc_table[NR_TASKS + NR_PROCS];

PUBLIC TASK task_table[NR_TASKS] = {
    {task_tty, STACK_SIZE_TTY, "tty"},
    {F, STACK_SIZE_F, "F"},
};

PUBLIC TASK user_proc_table[NR_PROCS] = {
    {ReaderA, STACK_SIZE_READER_A, "ReaderA"},
    {ReaderB, STACK_SIZE_READER_B, "ReaderB"},
    {ReaderC, STACK_SIZE_READER_C, "ReaderC"},
    {WriterD, STACK_SIZE_Writer_D, "WriterD"},
    {WriterE, STACK_SIZE_Writer_E, "WriterE"},
};

PUBLIC char task_stack[STACK_SIZE_TOTAL];

PUBLIC TTY tty_table[NR_CONSOLES];
PUBLIC CONSOLE console_table[NR_CONSOLES];

PUBLIC irq_handler irq_table[NR_IRQ];

PUBLIC system_call sys_call_table[NR_SYS_CALL] = {sys_get_ticks, sys_write,
                                                   sys_process_sleep, sys_P, sys_V};
```

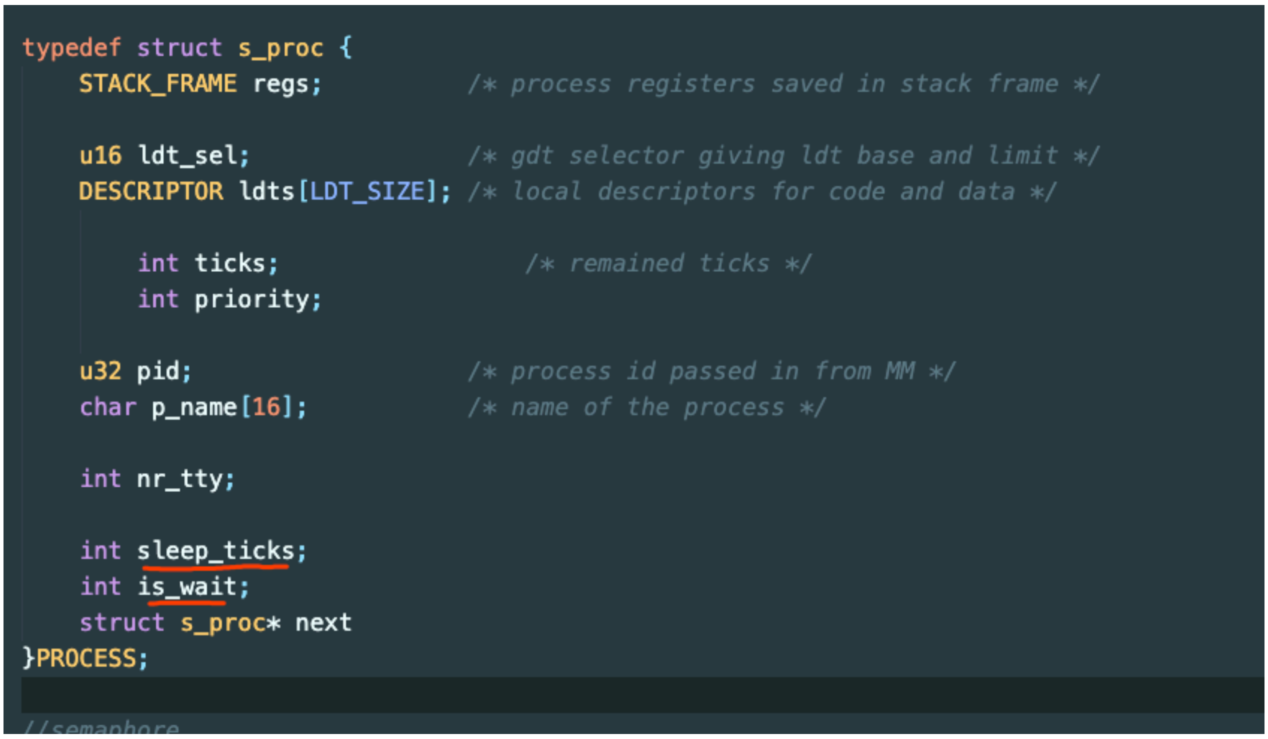
基于 `orange` 的实现，添加守护进程 `F` , 添加用户进程 `ReaderA` , `ReaderB`, `ReaderC` , `WriterD` , `WriterE`

并且添加系统调用 `sys_process_sleep` , `sys_P`, `sys_V` 三个系统调用。

- 用户进程和守护进程的定义都位于 `kernel/main.c`
- 系统调用的定义都位于 `kernel/proc.c`

2.2 数据模型修改

- 进程 `struct` 修改，为 `sys_process_sleep` 和PV操作的实现添加了 `sleep_ticks` 和 `is_wait` 的内容



```
typedef struct s_proc {
    STACK_FRAME regs;          /* process registers saved in stack frame */

    u16 ldt_sel;               /* gdt selector giving ldt base and limit */
    DESCRIPTOR ldts[LDT_SIZE]; /* local descriptors for code and data */

        int ticks;                 /* remained ticks */
        int priority;

    u32 pid;                   /* process id passed in from MM */
    char p_name[16];           /* name of the process */

    int nr_tty;

    int sleep_ticks;
    int is_wait;
    struct s_proc* next
}PROCESS;

//semaphore
```

- 添加信号量结构体

```
//semaphore
typedef struct semaphore {
  int value;
  PROCESS* queue
}SEMAPHORE;
```

2.3 代码逻辑添加

最主要的代码逻辑都位于 `kernel/proc.c` 当中，包括了 *PV* 操作的具体实现内容。此外，还改动了 `schedule` 调度逻辑，使其能够兼容进程睡眠（不占用时间片）的模式。