1. 包装系统调用

```
PUBLIC void sys_sleep(int milli_sec,int type)
{
    int ticks = milli_sec / 1000 * HZ * 10;  // 乘10是为了修正系统的时钟中断错误
    p_proc_ready->sleeping = ticks;
    p_proc_ready->issleep = type;
    schedule();
}

PUBLIC void sys_write_str(char* buf, int len)
{
    CONSOLE* p_con = console_table;
    for (int i = 0;i < len; i++){
        out_char(p_con, buf[i]);
    }
}
```

2. 信号量与PV操作

```
PUBLIC void p_proc(SEMAPHORE* s){
    disable_int();
    s->value--;
    if (s->value < 0){
        p_proc_ready->blocked = 1;
        s->p_list[s->tail] = p_proc_ready;
        s->tail = (s->tail + 1) % NR_PROCS;
        schedule();
    }
    enable_int();
}


PUBLIC void v_proc(SEMAPHORE* s){
    disable_int();
    s->value++;
    if (s->value <= 0){
        s->p_list[s->head]->blocked = 0;
        s->head = (s->head + 1) % NR_PROCS;
    }
    enable_int();
}
```

3. 读者写者问题

3.1 三种策略

分别设计读者优先、写者优先以及读写公平，通过表驱动来实现不同的策略

3.2 具体实现

```
void read_fair(char proc, int slices){
    P(&wait_mutex);
    P(&read_count_mutex);
    P(&read_mutex);
    if (reader_count == 0) {
        P(&read_write_mutex);
    }
    reader_count++;
    V(&read_mutex);
    V(&wait_mutex);
    read_proc(proc, slices);
    P(&read_mutex);
    reader_count--;
    if (reader_count == 0) {
        V(&read_write_mutex);
    }
    V(&read_mutex);
    V(&read_count_mutex);
}

void write_fair(char proc, int slices){
    P(&wait_mutex);
    P(&read_write_mutex);
    writing = 1;
    V(&wait_mutex);
    write_proc(proc, slices);
    writing = 0;
    V(&read_write_mutex);
}

void read_rfirst(char proc, int slices){
    P(&read_count_mutex);
    P(&read_mutex);
    if (reader_count==0) {
        P(&read_write_mutex);
    }
    reader_count++;
    V(&read_mutex);
    read_proc(proc, slices);
    P(&read_mutex);
    reader_count--;
    if (reader_count==0) {
        V(&read_write_mutex);
    }
    V(&read_mutex);
    V(&read_count_mutex);

}

void write_rfirst(char proc, int slices){
    P(&read_write_mutex);
    writing = 1;
    write_proc(proc, slices);
```

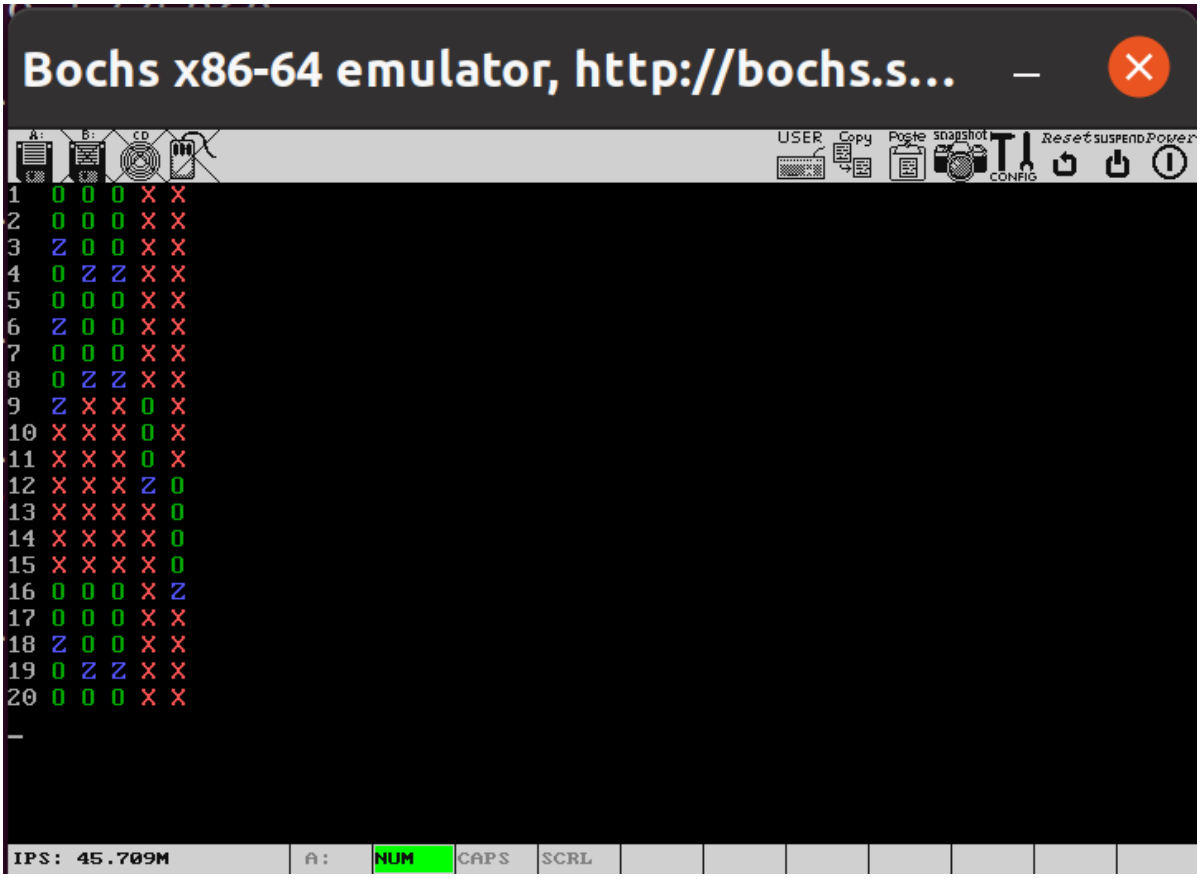
```
    writing = 0;
    V(&read_write_mutex);
}

void read_wfirst(char proc, int slices){
    P(&read_count_mutex);
    P(&wait_mutex);
    P(&read_mutex);
    if (reader_count==0) {
        P(&read_write_mutex);
    }
    reader_count++;
    V(&read_mutex);
    V(&wait_mutex);
    read_proc(proc, slices);
    P(&read_mutex);
    reader_count--;
    if (reader_count==0) {
        V(&read_write_mutex);
    }
    V(&read_mutex);
    V(&read_count_mutex);
}

void write_wfirst(char proc, int slices){
    P(&write_mutex);
    if (writers == 0) {
        P(&wait_mutex);
    }
    writers++;
    V(&write_mutex);
    P(&read_write_mutex);
    writing = 1;
    write_proc(proc, slices);
    writing = 0;
    V(&read_write_mutex);
    P(&write_mutex);
    writers--;
    if (writers == 0) {
        V(&wait_mutex);
    }
    V(&write_mutex);
}
```

3.3 reporter

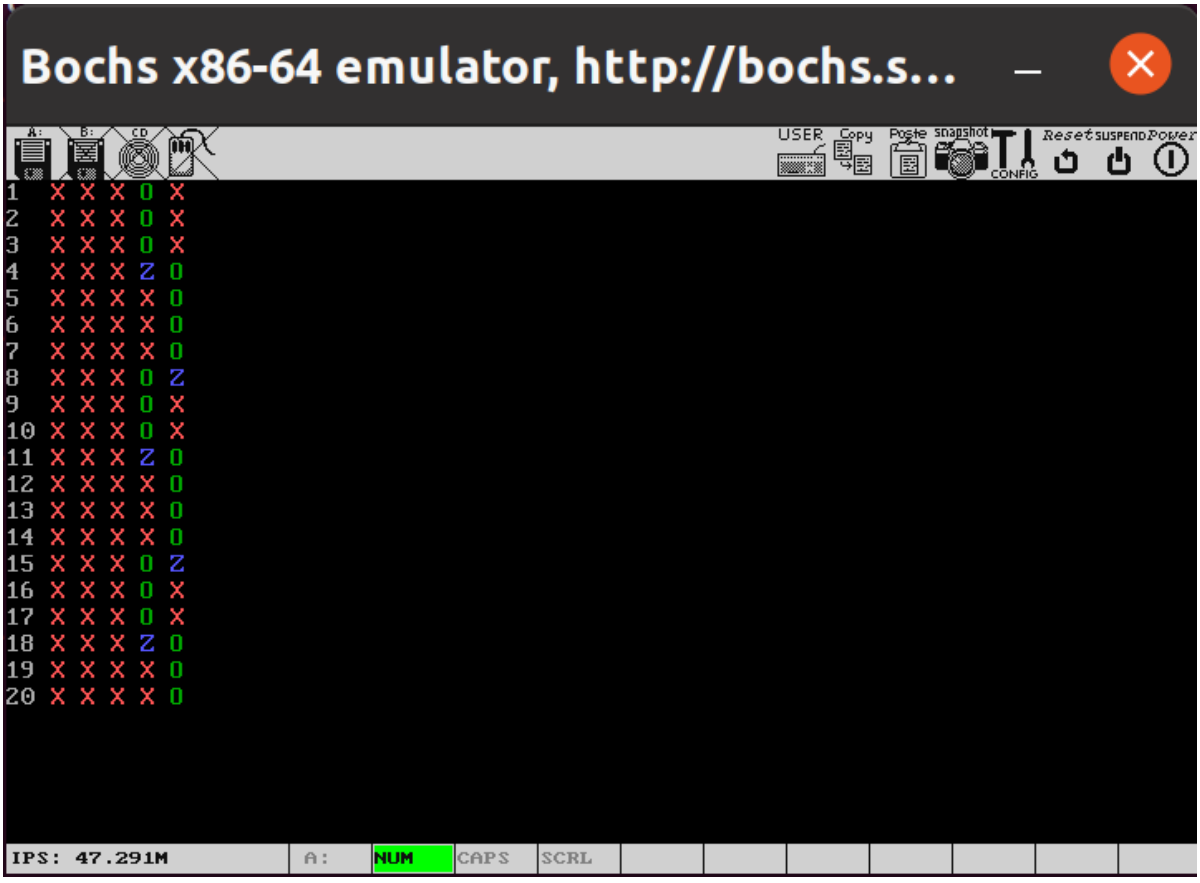
在每个时间片最后，reporter会打印出一行状态。将reporter在最开始时休息一个时间片再打印，每打印一次后休息一个时间片

3.4 运行结果

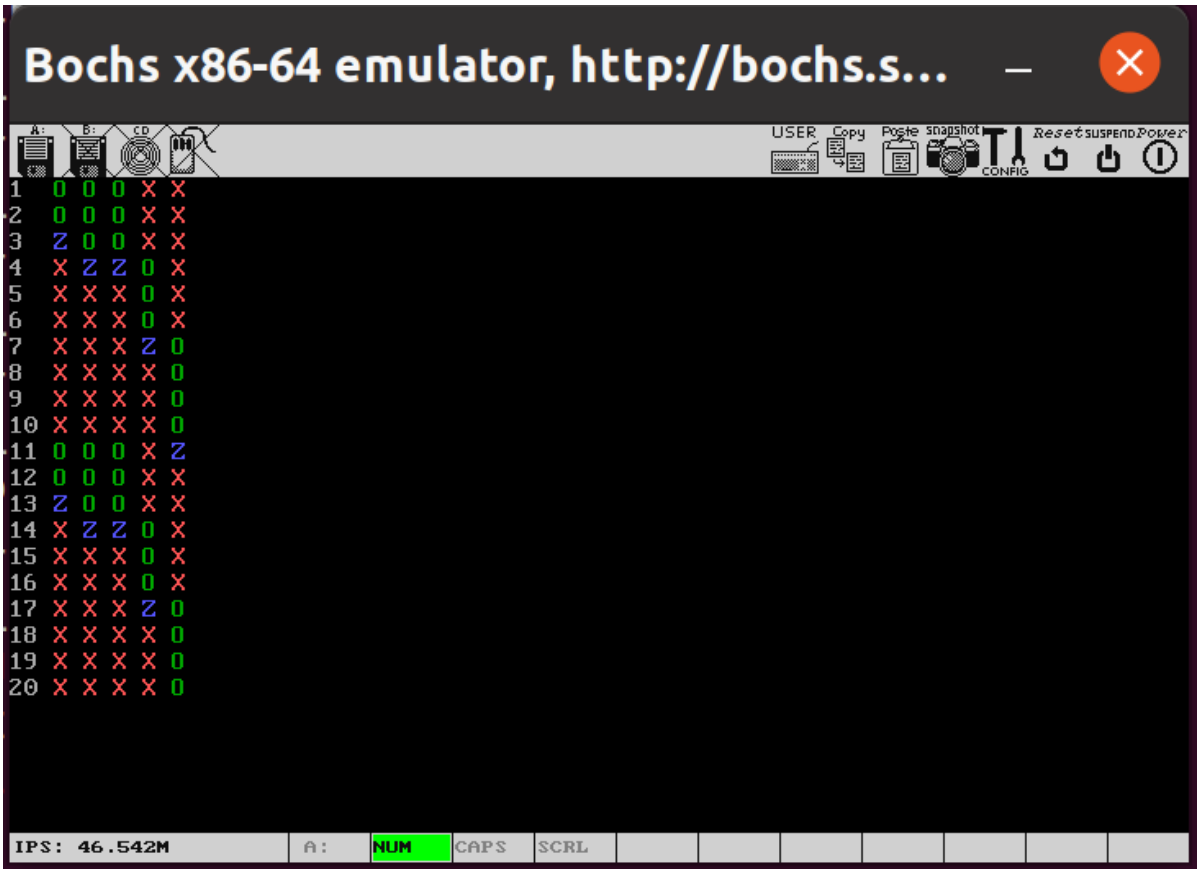
3.4.1 读者优先，最大3读者，休息1个时间片



3.4.2 写者优先，最大3读者，休息1个时间片



3.4.3 读写公平，最大3读者，休息1个时间片



4. 生产者消费者问题

4.1 信号量的设计

分别设计empty、count_A、count_B三个信号量表示空仓库数量、仓库中A产品数量、仓库中B产品数量

4.2 具体实现

```
void WriterE()
{
    while(1){
        P(&empty);
        write_proc();
        V(&count_A);
    }
}

void WriterF()
{
    while(1){
        P(&empty);
        write_proc();
        V(&count_B);
    }
}

void ReaderB()
{

    while(1){
```

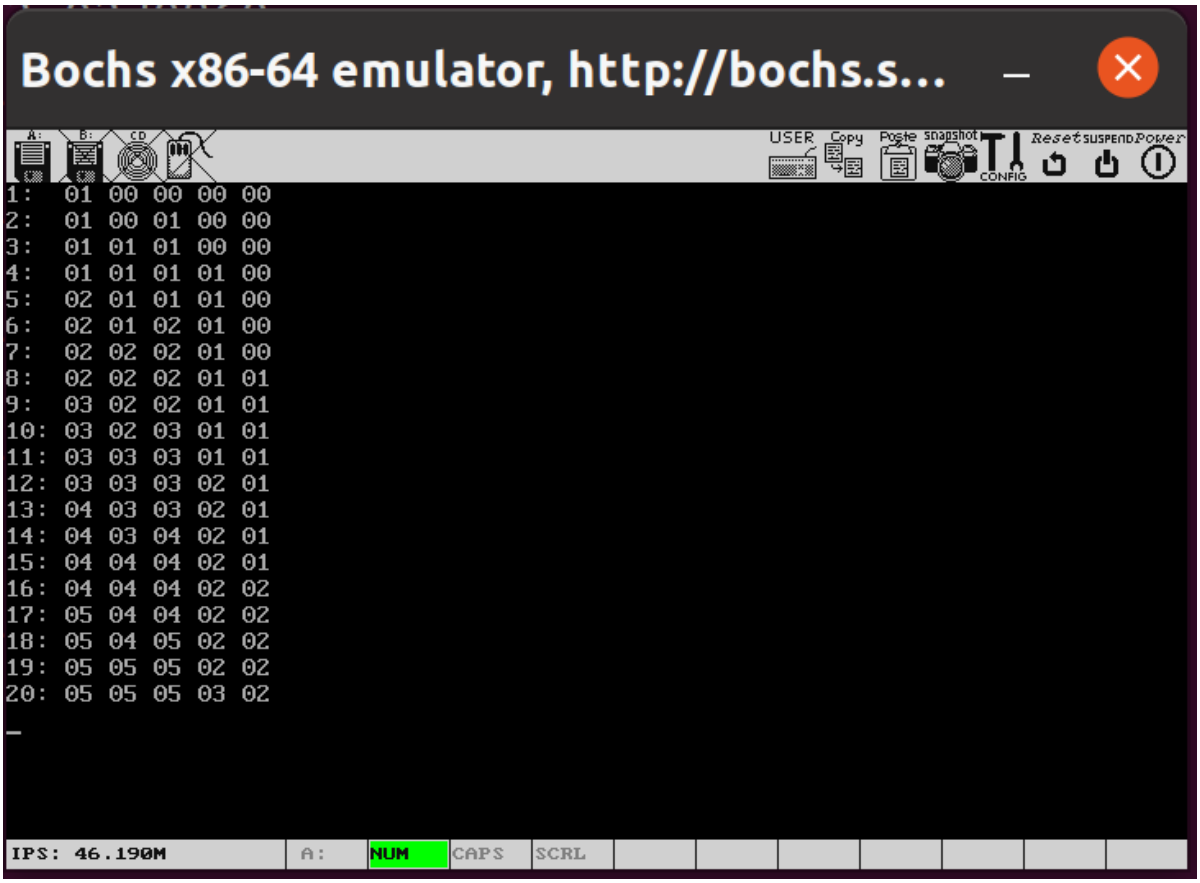
```
        P(&count_A);
        read_proc();
        V(&empty);
    }
}

void ReaderC()
{

    while(1){
        P(&count_B);
        read_proc();
        V(&empty);
    }
}

void ReaderD()
{

    while(1){
        P(&count_B);
        read_proc();
        V(&empty);
    }
}
```

4.3 reporter

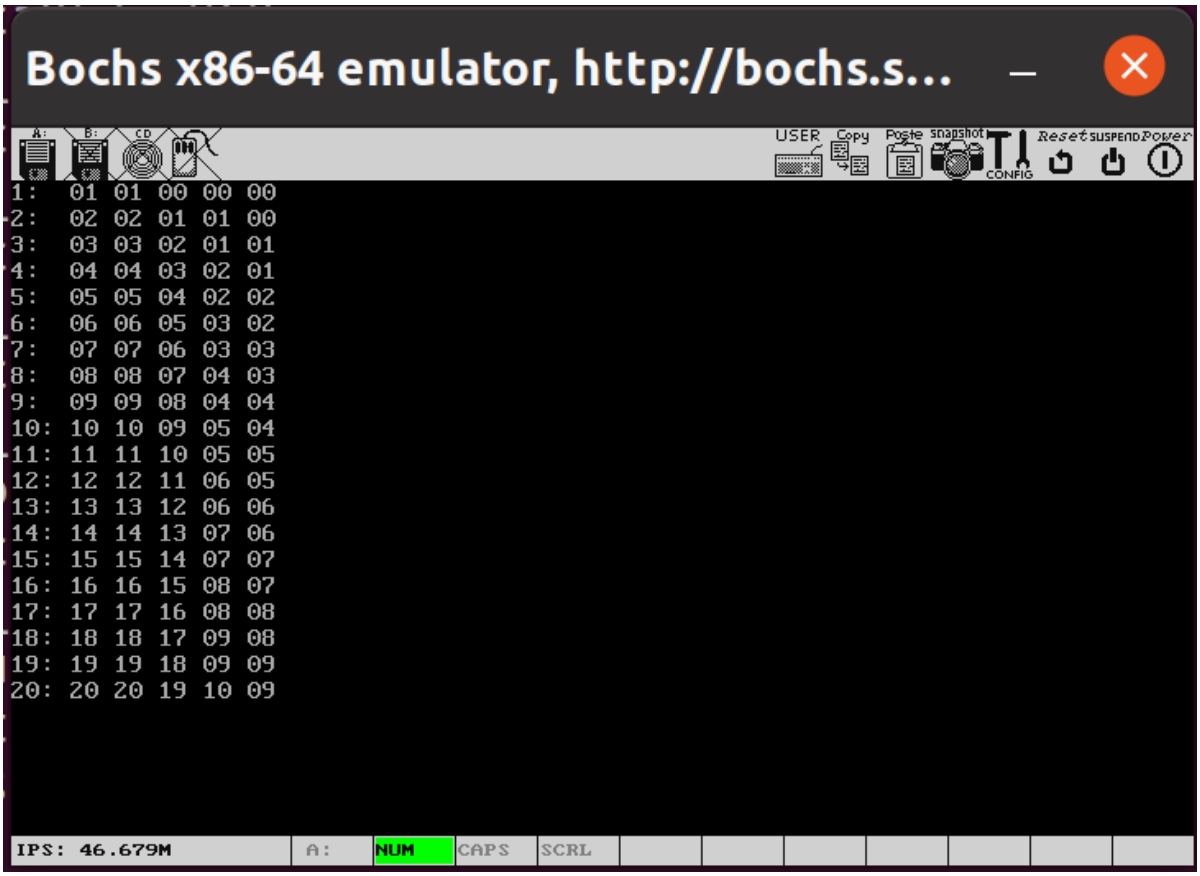
在每个时间片最后，reporter会打印出一行状态。将reporter在最开始时休息一个时间片再打印，每打印一次后休息一个时间片

4.4 运行结果

4.4.1 仓库为1时


Bochs x86-64 emulator, http://bochs.s...
1: 01 00 00 00 00
2: 01 00 01 00 00
3: 01 01 01 00 00
4: 01 01 01 01 00
5: 02 01 01 01 00
6: 02 01 02 01 00
7: 02 02 02 01 00
8: 02 02 02 01 01
9: 03 02 02 01 01
10: 03 02 03 01 01
11: 03 03 03 01 01
12: 03 03 03 02 01
13: 04 03 03 02 01
14: 04 03 04 02 01
15: 04 04 04 02 01
16: 04 04 04 02 02
17: 05 04 04 02 02
18: 05 04 05 02 02
19: 05 05 05 02 02
20: 05 05 05 03 02
IPS: 46.190M
A:
NUM
CAPS
SCRL

4.4.2 仓库为5时


Bochs x86-64 emulator, http://bochs.s...
1: 01 01 00 00 00
2: 02 02 01 01 00
3: 03 03 02 01 01
4: 04 04 03 02 01
5: 05 05 04 02 02
6: 06 06 05 03 02
7: 07 07 06 03 03
8: 08 08 07 04 03
9: 09 09 08 04 04
10: 10 10 09 05 04
11: 11 11 10 05 05
12: 12 12 11 06 05
13: 13 13 12 06 06
14: 14 14 13 07 06
15: 15 15 14 07 07
16: 16 16 15 08 07
17: 17 17 16 08 08
18: 18 18 17 09 08
19: 19 19 18 09 09
20: 20 20 19 10 09
IPS: 46.679M
A:
NUM
CAPS
SCRL