说明文档

201250203 张若皓

系统调用

不被分配时间片

```
PUBLIC void sys_sleep(int milli_sec)
{
  int ticks = milli_sec / 1000 * HZ * 10;
  p_proc_ready->sleeping = ticks;
    schedule();
}
```

输出

```
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]);
  }
}
```

读者写者问题实现

读者和写者进程

```
/*======================================================================*
                                ReaderB
 *======================================================================*/
void ReaderB()
{
  state[0] =0;
  sleep_ms(TIME_SLICE);
  while(1){
    read_funcs[strategy]('B', 2);
    sleep_ms(TIME_SLICE);
  }
}

/*======================================================================*
                                ReaderC
 *======================================================================*/
```

```
void ReaderC()
{
  state[1] =0;
  sleep_ms(2*TIME_SLICE);
  while(1){
    read_funcs[strategy]('C', 3);
    sleep_ms(TIME_SLICE);
  }
}

/*==========================================================================*
                                ReaderD
 *==========================================================================*/
void ReaderD()
{
  state[2] =0;
  sleep_ms(3*TIME_SLICE);
  while(1){
    read_funcs[strategy]('D',3);
    sleep_ms(TIME_SLICE);
  }
}

/*==========================================================================*
                                WriterE
 *==========================================================================*/
void WriterE()
{
  state[3] =0;
  sleep_ms(4*TIME_SLICE);
  while(1){
    write_funcs[strategy]('E', 3);
    sleep_ms(TIME_SLICE);
  }
}

/*==========================================================================*
                                WriterF
 *==========================================================================*/
void WriterF()
{
  state[4] = 0;
  sleep_ms(5*TIME_SLICE);
  while(1){
    write_funcs[strategy]('F', 4);
    sleep_ms(TIME_SLICE);
  }
}
```

用来输出状态的普通进程A

```
/*================================================================================*
                                Reporter
 *================================================================================*/
void ReporterA()
{
  sleep_ms(TIME_SLICE);
  int time=1;
  while(1){
    if(time>=10){
      if(time==21){
        break;
      }
      if(time==20){
        printf("%c%d",'\06',2);
        printf("%c%d ",'\06',0);
      }
      else{
        printf("%c%d",'\06',1);
        printf("%c%d ",'\06',time-10);
      }
    }
    else{
      printf("%c%d ",'\06',time);
    }

    time++;

    for (int i = 0; i < 5; i++) {
          if(i<4){
        if(state[i]==0){
          printf("%c%c ",'\03','Z');
        }
        else if(state[i]==1){
          printf("%c%c ",'\02','O');
        }
        else if(state[i]==2){
          printf("%c%c ",'\01','X');
        }
      }
      else{
        if(state[i]==0){
          printf("%c%c\n",'\03','Z');
        }
        else if(state[i]==1){
          printf("%c%c\n",'\02','O');
        }
        else if(state[i]==2){
```

```
            printf("%c%c\n",'\01','X');
          }
        }
      }
      sleep_ms(TIME_SLICE);
    }
    while(1){
      sleep_ms(TIME_SLICE);
    }
}
```

修改最大读者

在const.h中进行修改

```
//同时读的最大个数
#define MAX_READERS     3
```

修改进程休息时间

每个Reader和Writer中的sleep_ms函数中的参数进行修改。

分别实现读者优先和写者优先

在main.c中的strategy变量中进行修改

0，1，2分别对应 公平、读者优先、写者优先。

解决进程饿死

使用了公平读写进行完成。