OS lab6

PB21051012 刘祥辉

实验代码：

lab6写了优先级抢占调度和短作业优先调度。

task.c中有三个队列，分别为ready_queue,running_queue,waiting_queue;进程tcb存储在三个队列中的一个，队列出入管理程序有`tskEnqueueFCFS` `tskDequeueFCFS` `ReturnHead` `Enqueue` `Dequeue` 等函数接口，与上次实验类似

```
void tskDequeueFCFS(myTCB *tsk) {//rqFCFS出队
     if(rqFCFS.head==rqFCFS.tail){
          if(rqFCFS.head==NULL) return;
          else{
               rqFCFS.head = NULL;
               rqFCFS.tail = NULL;
          }
     }
     else{
          rqFCFS.head = rqFCFS.head->next;
     }
myTCB *ReturnHead(Queue *handler)
{
    if (handler->head == 0)
        return idleTask;
    else
        return handler->head;
}
void Enqueue(Queue *handler, myTCB *tsk)
{
    if (handler->head == 0)
    {
        handler->head = handler->tail = tsk;
        handler->tail->next = 0;
        return;
    }
    handler->tail->next = tsk;
    handler->tail = tsk;
    tsk->next = 0;
    return;
}
myTCB *Dequeue(Queue *handler)
{
    myTCB *TCB = handler->head;
    if (handler->head == 0)
        return TCB;
    if (handler->head == handler->tail)
    {
        handler->head = handler->tail = 0;
        return TCB;
    }
    handler->head = (handler->head)->next;
    return TCB;
}
```

每次时钟秒数增加时候调用`Queue_manager`函数

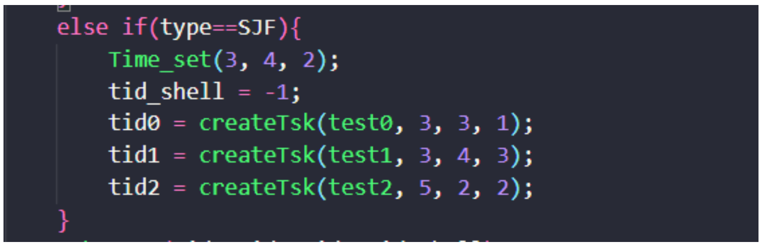
```
void oneTickUpdateWallClock(void){
    ms += 10;  // ?100HZ?
        if (ms>=1000) {
                        ms=0;
                        ss++;
            InterruptTime++;
                        Queue_manager();
                        }
        if (ss>=60) {ss=0;mm++;}
        if (mm>=60) {mm=0;hh++;}
        if (hh>=24) hh=0;

    if (wallClock_hook) wallClock_hook();
}
```

```
void Queue_manager(){
    myTCB *TCB = 0;
    if (is_schedule == 0)
        return;
    if (currentTask != 0 && currentTask != idleTask && currentTask->exeTime > 0)
        currentTask->exeTime--;

    do{
        TCB = ReturnHead(Handler_dready);
        if (TCB == idleTask)
            break;
        if (TCB->arrTime <= InterruptTime)
        {
            Dequeue(Handler_dready);
            Enqueue(Handler_ready, TCB);
            QueueSort(Handler_ready, type);
        }
    } while (TCB->arrTime <= InterruptTime);
    myTCB *nextTCB = ReturnHead(Handler_ready);

    if (nextTCB == idleTask)
        return;

    if(type==PRIO){
        if (currentTask->priority > nextTCB->priority)
        {
            unsigned long **p = &BspContext;

            QueueDestroy(Handler_run, currentTask->tid);
            Dequeue(Handler_ready);
            Enqueue(Handler_run, nextTCB);
            Enqueue(Handler_ready, currentTask);
            QueueSort(Handler_ready, type);

            myTCB *temp = currentTask;

            currentTask = nextTCB;
```

```
                context_switch(&BspContext, currentTask->stack_top);
                currentTask = Dequeue(Handler_ready);
                Enqueue(Handler_run, currentTask);
            }
        }
        return;
    }
```

每当时间过了一秒，如果当前是抢占式调度，比较running_queue和waiting_queue的优先级，如果达到抢占要求则改变队列，以达到抢占调度的效果。

SJU



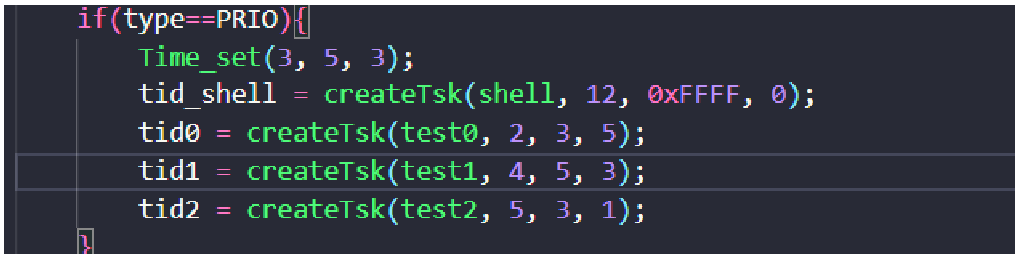
```
else if(type==SJF){
    Time_set(3, 4, 2);
    tid_shell = -1;
    tid0 = createTsk(test0, 3, 3, 1);
    tid1 = createTsk(test1, 3, 4, 3);
    tid2 = createTsk(test2, 5, 2, 2);
}
```

各任务到达时间与运行时间：

test0，arr_time:3；exe_time:3； test1，arr_time:3，exe_time:4； test2，arr_time:5，exe_time:2；

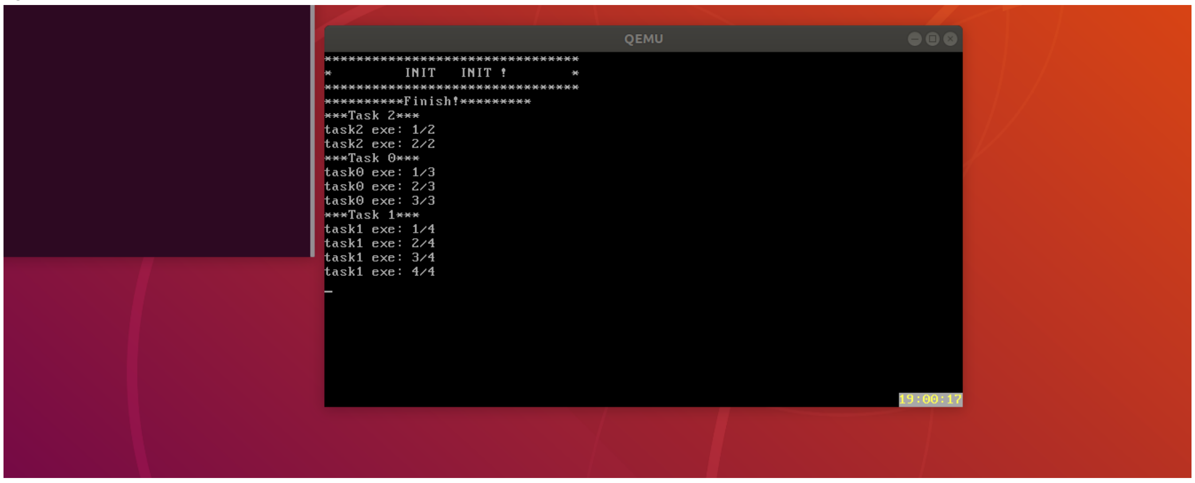
预期运行：
0~3 idle;
3~6 tsk0;
6~8 tsk2;
8~12 tsk1;

PRIO



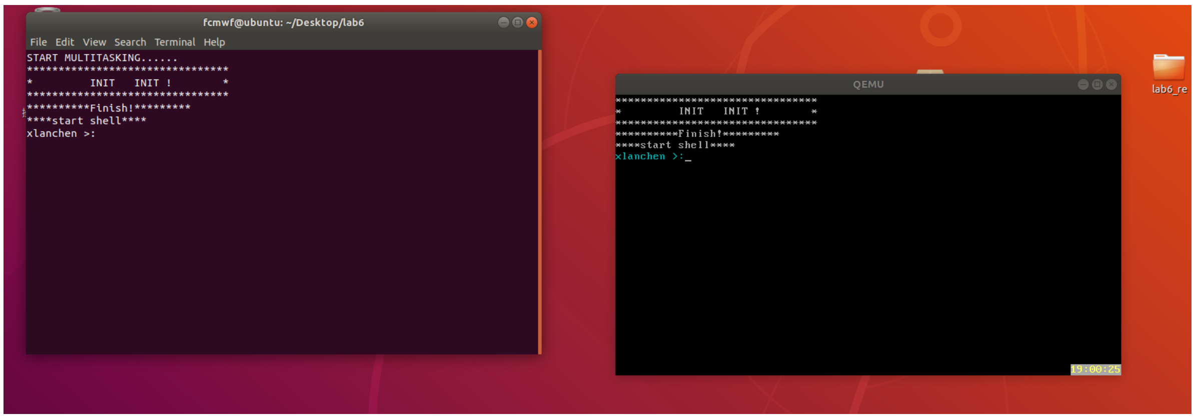
```
if(type==PRIO){
    Time_set(3, 5, 3);
    tid_shell = createTsk(shell, 12, 0xFFFF, 0);
    tid0 = createTsk(test0, 2, 3, 5);
    tid1 = createTsk(test1, 4, 5, 3);
    tid2 = createTsk(test2, 5, 3, 1);
}
```

shell，arr_time:12，prio:0，exe_time:0xFFFF；
test0，arr_time:2，prio:5，exe_time:3；
test1，arr_time:4，prio:3，exe_time:5；
test2，arr_time:5，prio:1，exe_time:3；
预期运行
0~2 idle;
2~5 tsk0;
5~10 tsk1;
10~13 tsk2;

运行结果：
SJU：


```
********************************
*          INIT    INIT !          *
********************************
**********Finish!*********
***Task 2***
task2 exe: 1/2
task2 exe: 2/2
***Task 0***
task0 exe: 1/3
task0 exe: 2/3
task0 exe: 3/3
***Task 1***
task1 exe: 1/4
task1 exe: 2/4
task1 exe: 3/4
task1 exe: 4/4
_
```

```
19:00:17
```

PRIO：

fcmwf@ubuntu: ~/Desktop/lab6
File Edit View Search Terminal Help
QEMU
lab6_re

```
START MULTITASKING......
********************************
*         INIT   INIT !        *
********************************
**********Finish!*********
****start shell****
xlanchen >:
```

```
********************************
*         INIT   INIT !        *
********************************
**********Finish!*********
****start shell****
xlanchen >:_
```

```
19:00:25
```

优先级调度不知道什么情况tsk0,tsk1,tsk2直接跳过了，直接执行shell任务了，临近ddl没时间debug了，TvT