

# LAB6 report

毛陈诚 PB20111694

## 1. 非抢占式SJF

样例一：如图工作时间为task2>task0>task1, 因为非抢占式, 所以实际运行顺序task1>task0>task2

```
doSomeTestBefore();

myPrintf(0x07, "*****");
myPrintf(0x07, "*****");
myPrintf(0x07, "*****");

createTsk(myTsk0, (tskPara){ 3, 10, 0});
createTsk(myTsk1, (tskPara) { 2, 2, 0 });
createTsk(myTsk2, (tskPara) { 1, 21, 0 });
```

```
Machine View
*****
INIT INIT !
*****
Tsk1: HELLO WORLD!
*****
Tsk0: HELLO WORLD!
*****
Tsk2: HELLO WORLD!
*****
```

样例二：图工作时间为task1>task0>task2, 因为非抢占式, 所以实际运行顺序task2>task0>task1

```
myPrintf(0x07, "*****");
myPrintf(0x07, "*****");
myPrintf(0x07, "*****");

createTsk(myTsk0, (tskPara){ 3, 8, 0});
createTsk(myTsk1, (tskPara) { 2, 21, 0 });
createTsk(myTsk2, (tskPara) { 1, 6, 0 });
```

```
Machine View
*****
INIT INIT !
*****
Tsk2: HELLO WORLD!
*****
Tsk0: HELLO WORLD!
*****
Tsk1: HELLO WORLD!
*****
```

## 2. 抢占式Priority 调度

样例一：如图优先级为task2>task1>task0。实际运行时从task1开始, 在第5s时task0到达, 因为task0优先级比task1低, 所以不抢占, 在第6s时task2到达, 由于task2优先级比task1高, 所以发生抢占, 所以执行次序为task2, task1, task0, 满足要求

```
myPrintf(0x07, "*****");
myPrintf(0x07, "*****");
myPrintf(0x07, "*****");

createTsk(myTsk0, (tskPara){ 3, 10, 5});
createTsk(myTsk1, (tskPara) { 2, 22, 0 });
createTsk(myTsk2, (tskPara) { 1, 8, 6 });

//#error "TODO: 初始化 shell 并创建 shell tas
```

```
QEMU
Machine View
*****
INIT INIT !
*****
xlanchen >: *****
Tsk2: HELLO WORLD!
*****
Tsk1: HELLO WORLD!
*****
Tsk0: HELLO WORLD!
*****
```

样例二：

2. 如图优先级为task2>task1>task0。实际运行时在第2秒从task0开始, 在第7s时task1到达, 因为task1优先级比task0高, 所以抢占, 在第11s时task1结束, 重新由运行task0, 在第12秒, 由于task2优先级比task0高, 所以发生抢占, 所以执行次序为task1, task2, task0, 满足要求

```
doSomeTestBefore();

myPrintf(0x07, "*****");
myPrintf(0x07, "*****");
myPrintf(0x07, "*****");

createTsk(myTsk0, (tskPara){ 3, 13, 2});
createTsk(myTsk1, (tskPara) { 2, 11, 7 });
createTsk(myTsk2, (tskPara) { 1, 13, 12 });
```

```
QEMU
Machine View
*****
INIT INIT !
*****
Tsk1: HELLO WORLD!
*****
Tsk2: HELLO WORLD!
*****
Tsk0: HELLO WORLD!
*****
```

