

廈門大學



软件学院

《实用操作系统》Project3

题 目 向 LiteOS 中添加一个短作业优先调度策略

姓 名 陈澄

学 号 32420212202930

班 级 软工三班

实验时间 2023/9/26

2023 年 09 月 26 日

1 实验目的

向 LiteOS 中添加一个简单的基于线程运行时的短作业优先调度策略

2 实验环境

主机：Windows 11

虚拟机：Ubuntu 18.04

开发板：IMAX6ULL MINI

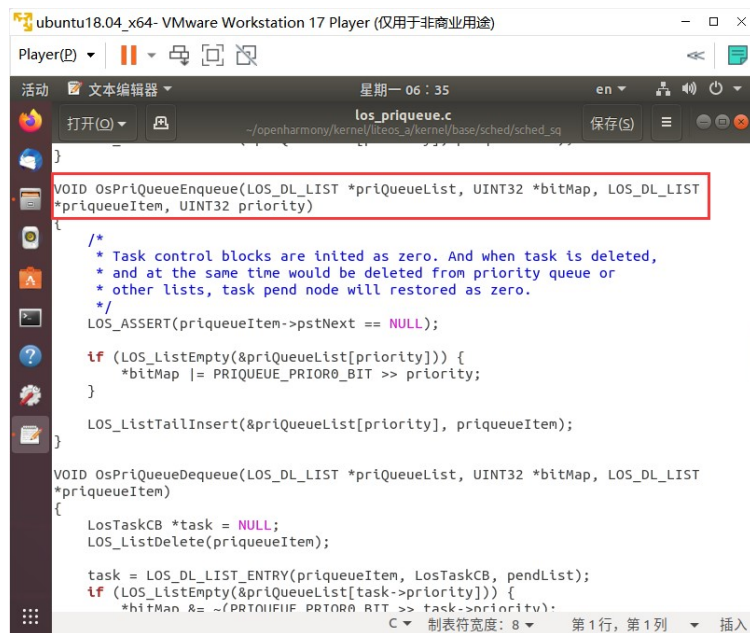
终端：MobaXterm

3 实验内容

1. 修改优先级队列函数

打开 openharmony/kernel/liteos_a/kernel/base/sched/sched_sq/los_pqueue.c

找到 OsPriQueueEnqueue 方法



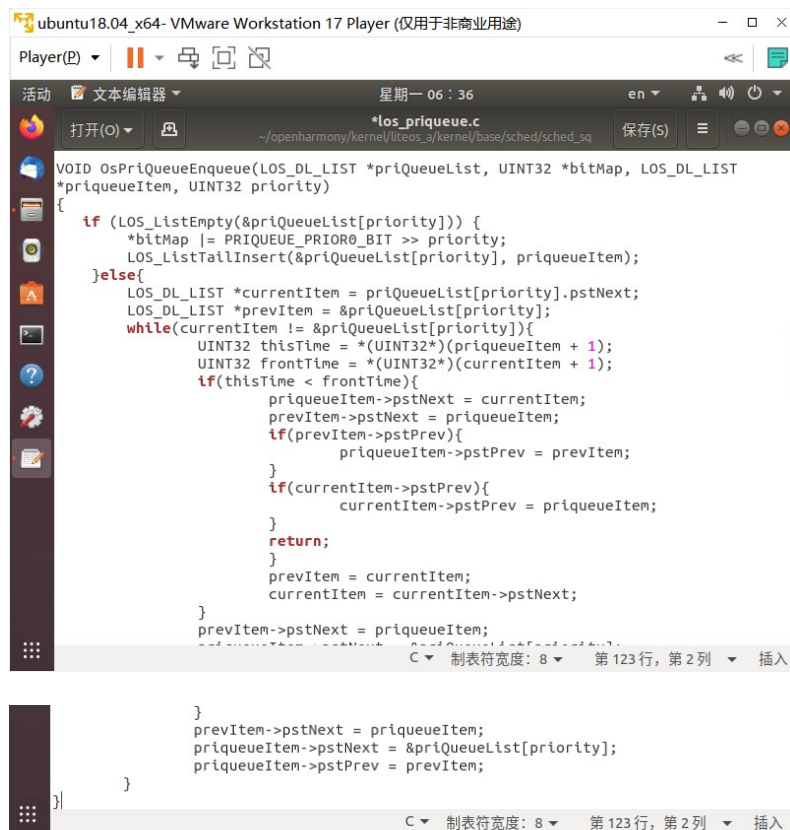
```
ubuntu18.04_x64- VMware Workstation 17 Player (仅用于非商业用途)
Player(P) | 文本编辑器 | 星期一 06:35 | en | 保存(S) | 插入
~/openharmony/kernel/liteos_a/kernel/base/sched/sched_sq/los_pqueue.c
}
VOID OsPriQueueEnqueue(LOS_DL_LIST *priQueueList, UINT32 *bitMap, LOS_DL_LIST
*priQueueItem, UINT32 priority)
{
    /*
     * Task control blocks are initied as zero. And when task is deleted,
     * and at the same time would be deleted from priority queue or
     * other lists, task pend node will restored as zero.
     */
    LOS_ASSERT(priQueueItem->pstNext == NULL);
    if (LOS_ListEmpty(&priQueueList[priority])) {
        *bitMap |= PRIQUEUE_PRIOR0_BIT >> priority;
    }
    LOS_ListTailInsert(&priQueueList[priority], priQueueItem);
}
VOID OsPriQueueDequeue(LOS_DL_LIST *priQueueList, UINT32 *bitMap, LOS_DL_LIST
*priQueueItem)
{
    LosTaskCB *task = NULL;
    LOS_ListDelete(priQueueItem);
    task = LOS_DL_LIST_ENTRY(priQueueItem, LosTaskCB, pendList);
    if (LOS_ListEmpty(&priQueueList[task->priority])) {
        *bitMap &= ~(PRIQUEUE_PRIOR0_BIT >> task->priority);
    }
}
```

该方法就是优先级队列函数

将其更改即可改变调度策略

本实验将其替换为运行时间短的优先策略

代码如下



```
ubuntu18.04_x64- VMware Workstation 17 Player (仅用于非商业用途)
Player(P)  星期一 06:36
*los_priqueue.c
~/openharmy/kernel/liteos_a/kernel/base/sched/sched_sq  保存(S)

VOID OsPriQueueEnqueue(LOS_DL_LIST *priQueueList, UINT32 *bitMap, LOS_DL_LIST
*priqueueItem, UINT32 priority)
{
    if (LOS_ListEmpty(&priQueueList[priority])) {
        *bitMap |= PRIQUEUE_PRIOR0_BIT >> priority;
        LOS_ListTailInsert(&priQueueList[priority], priqueueItem);
    }else{
        LOS_DL_LIST *currentItem = priQueueList[priority].pstNext;
        LOS_DL_LIST *prevItem = &priQueueList[priority];
        while(currentItem != &priQueueList[priority]){
            UINT32 thisTime = *(UINT32*)(priqueueItem + 1);
            UINT32 frontTime = *(UINT32*)(currentItem + 1);
            if(thisTime < frontTime){
                priqueueItem->pstNext = currentItem;
                prevItem->pstNext = priqueueItem;
                if(prevItem->pstPrev){
                    priqueueItem->pstPrev = prevItem;
                }
                if(currentItem->pstPrev){
                    currentItem->pstPrev = priqueueItem;
                }
                return;
            }
            prevItem = currentItem;
            currentItem = currentItem->pstNext;
        }
        prevItem->pstNext = priqueueItem;
        priqueueItem->pstNext = &priQueueList[priority];
        priqueueItem->pstPrev = prevItem;
    }
}
```

2. 修改 LosTaskCB 结构体

打开 openharmy/kernel/liteos_a/kernel/base/include/los_task_pri.h

找到 LosTaskCB 结构体

```
#define OS_RESOURCE_EVENT_FREE 0x04
#define OS_TCB_NAME_LEN 32

typedef struct {
    VOID *stackPointer; /*< Task stack pointer */
    UINT16 taskStatus; /*< Task status */
    UINT16 priority; /*< Task priority */
    UINT16 policy;
    UINT16 timeSlice; /*< Remaining time slice */
    UINT32 stackSize; /*< Task stack size */
    UINTPTR topOfStack; /*< Task stack top */
    UINT32 taskID; /*< Task ID */
    TSK_ENTRY_FUNC taskEntry; /*< Task entrance function */
    VOID *joinRetVal; /*< pthread adaption */
    VOID *taskSem; /*< Task-held semaphore */
    VOID *taskMux; /*< Task-held mutex */
    VOID *taskEvent; /*< Task-held event */
    UINTPTR args[4]; /*< Parameter, of which the maximum
    number is 4 */
    CHAR taskName[OS_TCB_NAME_LEN]; /*< Task name */
    LOS_DL_LIST pendList; /*< Task pend node */
    LOS_DL_LIST threadList; /*< thread list */
    SortLinkList sortList; /*< Task sortlink node */
    UINT32 eventMask; /*< Event mask */
    UINT32 eventMode; /*< Event mode */
    UINT32 priBitMap; /*< BitMap for recording the change of
    task priority,
    the priority can not be greater than
```

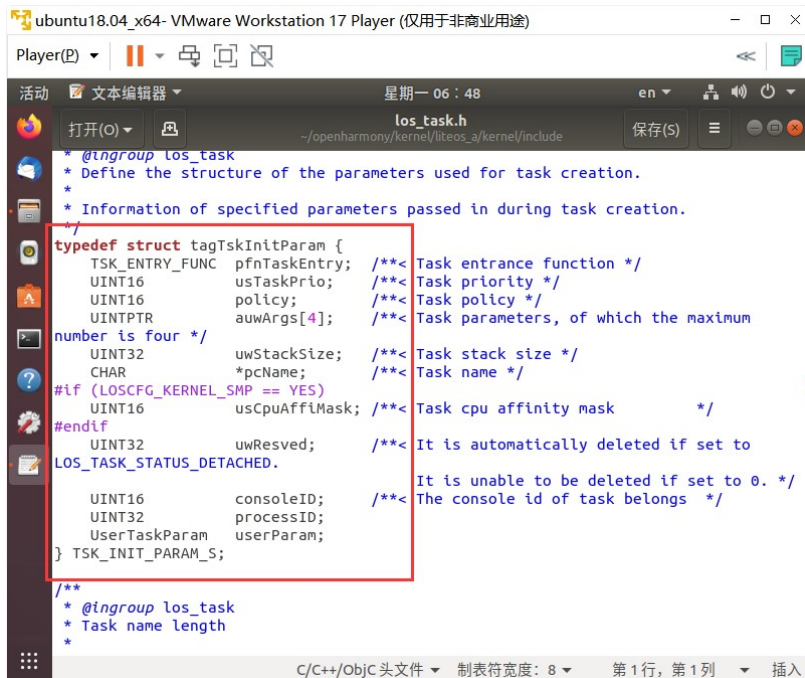
在 pendList 变量下添加一条 UINT32 RunTime

```
typedef struct {
    VOID *stackPointer; /*< Task stack pointer */
    UINT16 taskStatus; /*< Task status */
    UINT16 priority; /*< Task priority */
    UINT16 policy;
    UINT16 timeSlice; /*< Remaining time slice */
    UINT32 stackSize; /*< Task stack size */
    UINTPTR topOfStack; /*< Task stack top */
    UINT32 taskID; /*< Task ID */
    TSK_ENTRY_FUNC taskEntry; /*< Task entrance function */
    VOID *joinRetVal; /*< pthread adaption */
    VOID *taskSem; /*< Task-held semaphore */
    VOID *taskMux; /*< Task-held mutex */
    VOID *taskEvent; /*< Task-held event */
    UINTPTR args[4]; /*< Parameter, of which the maximum
    number is 4 */
    CHAR taskName[OS_TCB_NAME_LEN]; /*< Task name */
    LOS_DL_LIST pendList; /*< Task pend node */
    LOS_DL_LIST threadList; /*< thread list */
    UINT32 RunTime;
    SortLinkList sortList; /*< Task sortlink node */
    UINT32 eventMask; /*< Event mask */
    UINT32 eventMode; /*< Event mode */
    UINT32 priBitMap; /*< BitMap for recording the change of
    task priority,
    the priority can not be greater than
    31 */
```

3. 修改 TSK_INIT_PARAM_S 结构体

打开 openharmony/kernel/liteos_a/kernel/include/los_task.h

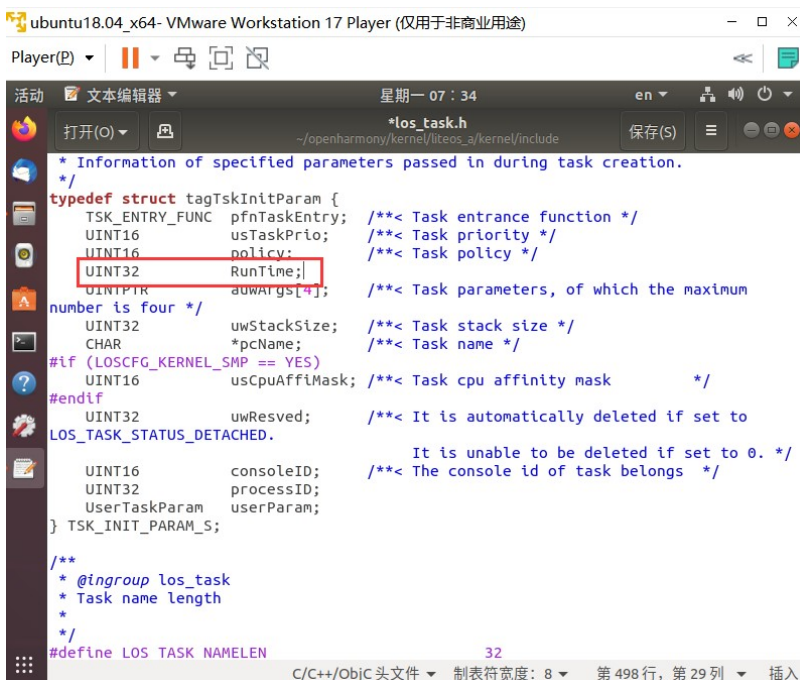
找到 TSK_INIT_PARAM_S 结构体



```
/* @ingroup los_task
 * Define the structure of the parameters used for task creation.
 * Information of specified parameters passed in during task creation.
 */
typedef struct tagTaskInitParam {
    TSK_ENTRY_FUNC pfnTaskEntry; /**< Task entrance function */
    UINT16 usTaskPrio; /**< Task priority */
    UINT16 policy; /**< Task policy */
    UINTPTR auwArgs[4]; /**< Task parameters, of which the maximum
number is four */
    UINT32 uwStackSize; /**< Task stack size */
    CHAR *pcName; /**< Task name */
    #if (LOSCFG_KERNEL_SMP == YES)
    UINT16 usCpuAffiMask; /**< Task cpu affinity mask */
    #endif
    UINT32 uwResved; /**< It is automatically deleted if set to
LOS_TASK_STATUS_DETACHED.
It is unable to be deleted if set to 0. */
    UINT16 consoleID; /**< The console id of task belongs */
    UINT32 processID;
    UserTaskParam userParam;
} TSK_INIT_PARAM_S;

/**
 * @ingroup los_task
 * Task name length
 */
```

在 policy 下添加一条 UINT32 RunTime



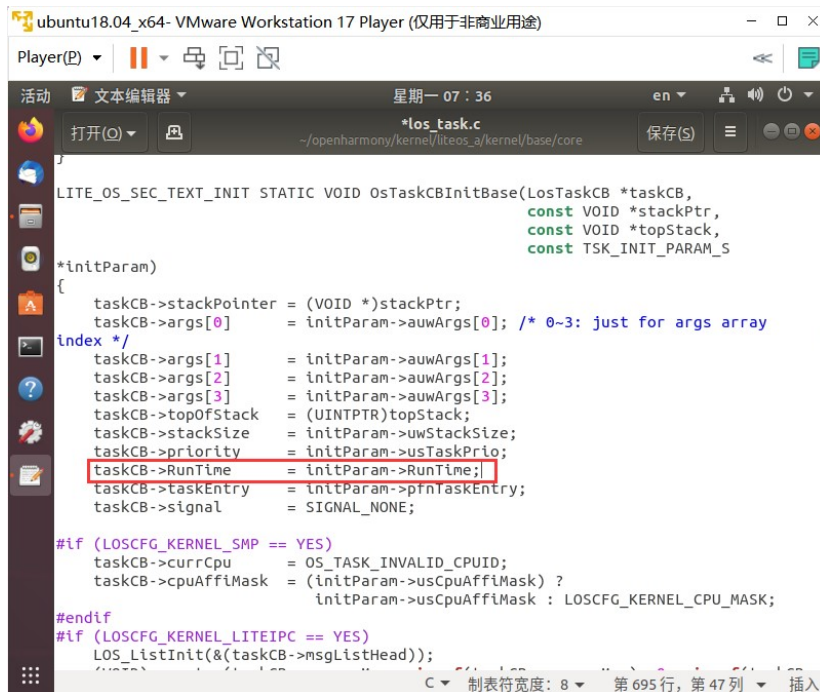
```
/* Information of specified parameters passed in during task creation.
 */
typedef struct tagTaskInitParam {
    TSK_ENTRY_FUNC pfnTaskEntry; /**< Task entrance function */
    UINT16 usTaskPrio; /**< Task priority */
    UINT16 policy; /**< Task policy */
    UINT32 RunTime;
    UINTPTR auwArgs[4]; /**< Task parameters, of which the maximum
number is four */
    UINT32 uwStackSize; /**< Task stack size */
    CHAR *pcName; /**< Task name */
    #if (LOSCFG_KERNEL_SMP == YES)
    UINT16 usCpuAffiMask; /**< Task cpu affinity mask */
    #endif
    UINT32 uwResved; /**< It is automatically deleted if set to
LOS_TASK_STATUS_DETACHED.
It is unable to be deleted if set to 0. */
    UINT16 consoleID; /**< The console id of task belongs */
    UINT32 processID;
    UserTaskParam userParam;
} TSK_INIT_PARAM_S;

/**
 * @ingroup los_task
 * Task name length
 */
#define LOS_TASK_NAMELEN 32
```

4. 修改任务赋值（初始化）函数

打开 openharmony/kernel/liteos_a/kernel/base/core/los_task.c

在 taskCB->priority 下添加一条：taskCB->RunTime = initParam->RunTime

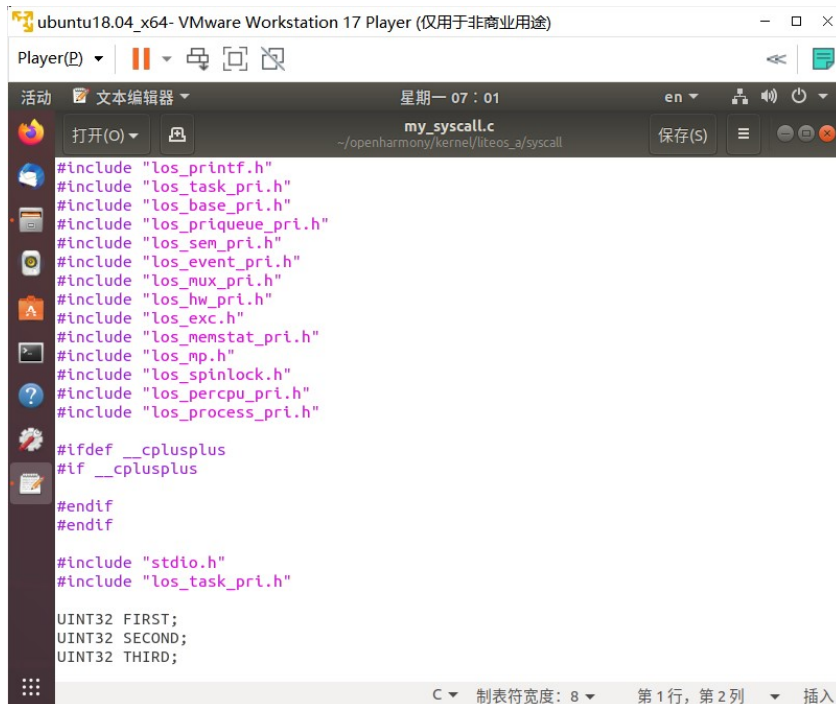


```
ubuntu18.04_x64- VMware Workstation 17 Player (仅用于非商业用途)
Player(P) | 活动 | 文本编辑器 | 星期一 07:36 | en | 保存(S) | 插入
~/openharmy/kernel/liteos_a/kernel/base/core
*los_task.c
LITE_OS_SEC_TEXT_INIT STATIC VOID OsTaskCBInitBase(LosTaskCB *taskCB,
                                                    const VOID *stackPtr,
                                                    const VOID *topStack,
                                                    const TSK_INIT_PARAM_S
                                                    *initParam)
{
    taskCB->stackPointer = (VOID *)stackPtr;
    taskCB->args[0] = initParam->auwArgs[0]; /* 0-3: just for args array
index */
    taskCB->args[1] = initParam->auwArgs[1];
    taskCB->args[2] = initParam->auwArgs[2];
    taskCB->args[3] = initParam->auwArgs[3];
    taskCB->topOfStack = (UINTPTR)topStack;
    taskCB->stackSize = initParam->uwStackSize;
    taskCB->priority = initParam->usTaskPrio;
    taskCB->RunTime = initParam->RunTime;
    taskCB->taskEntry = initParam->pfnTaskEntry;
    taskCB->signal = SIGNAL_NONE;

    #if (LOSCFG_KERNEL_SMP == YES)
    taskCB->currCpu = OS_TASK_INVALID_CPUID;
    taskCB->cpuAffiMask = (initParam->usCpuAffiMask) ?
        initParam->usCpuAffiMask : LOSCFG_KERNEL_CPU_MASK;
    #endif
    #if (LOSCFG_KERNEL_LITEIPC == YES)
    LOS_ListInit(&(taskCB->msgListHead));
    (taskCB->msgListHead) = (LOS_LIST_HEAD);
    #endif
}
```

5. 编写验证测试程序

更改 Project1 中制作的 openharmony/kernel/liteos_a/syscall/my_syscall.c



```
ubuntu18.04_x64- VMware Workstation 17 Player (仅用于非商业用途)
Player(P) | 活动 | 文本编辑器 | 星期一 07:01 | en | 保存(S) | 插入
~/openharmy/kernel/liteos_a/syscall
my_syscall.c
#include "los_printf.h"
#include "los_task_pri.h"
#include "los_base_pri.h"
#include "los_priqueue_pri.h"
#include "los_sem_pri.h"
#include "los_event_pri.h"
#include "los_mux_pri.h"
#include "los_hw_pri.h"
#include "los_exc.h"
#include "los_memstat_pri.h"
#include "los_mp.h"
#include "los_spinlock.h"
#include "los_percpu_pri.h"
#include "los_process_pri.h"

#ifdef __cplusplus
if __cplusplus
#endif
#endif

#include "stdio.h"
#include "los_task_pri.h"

UINT32 FIRST;
UINT32 SECOND;
UINT32 THIRD;
```


ubuntu18.04_x64- VMware Workstation 17 Player (仅用于非商业用途)

Player(P) | 活动 | 文本编辑器 | 星期一 08:15 | en | 保存(S) | 制表符宽度: 8 | 第 43 行, 第 28 列 | 插入

```
#define PRIOR 1

void PRIOR_FIRST_TASK(VOID)
{
    PRINTK("Task:One , RunTime:15 s\r\n");
    return;
}

void PRIOR_SECOND_TASK(VOID)
{
    PRINTK("Task:Two , RunTime:30 s\r\n");
    return;
}

void PRIOR_THIRD_TASK(VOID)
{
    PRINTK("Task:Three |, RunTime:10 s\r\n");
    return;
}

void MySyscall(int num){
    UINT32 ret;
    TSK_INIT_PARAM_S initParam;
    LOS_TaskLock();
    PRINTK("\nLOS_TaskLock() Success!\r\n\r\n");

    initParam.pfnTaskEntry = (TSK_ENTRY_FUNC)PRIOR_FIRST_TASK;
    initParam.usTaskPrio = PRIOR;
    initParam.pcName = "PRIOR_FIRST_TASK";
    initParam.uwStackSize = OS_TASK_RESOURCE_STATCI_SIZE;
```

ubuntu18.04_x64- VMware Workstation 17 Player (仅用于非商业用途)

Player(P) | 活动 | 文本编辑器 | 星期一 07:02 | en | 保存(S) | 制表符宽度: 8 | 第 1 行, 第 2 列 | 插入

```
initParam.RunTime = 15;
initParam.uwResved = LOS_TASK_STATUS_DETACHED;
ret = LOS_TaskCreate(&FIRST,&initParam);
if(ret != LOS_OK)
{
    LOS_TaskUnlock();
    PRINTK("Failed_1!\r\n");
    return;
}
PRINTK("Init_Task1_Success!\r\n");

initParam.pfnTaskEntry = (TSK_ENTRY_FUNC)PRIOR_SECOND_TASK;
initParam.usTaskPrio = PRIOR;
initParam.pcName = "PRIOR_SECOND_TASK";
initParam.uwStackSize = OS_TASK_RESOURCE_STATCI_SIZE;
initParam.RunTime = 30;
initParam.uwResved = LOS_TASK_STATUS_DETACHED;
ret = LOS_TaskCreate(&SECOND,&initParam);
if(ret != LOS_OK)
{
    LOS_TaskUnlock();
    PRINTK("Failed_2!\r\n");
    return;
}
PRINTK("Init_Task2_Success!\r\n");
```

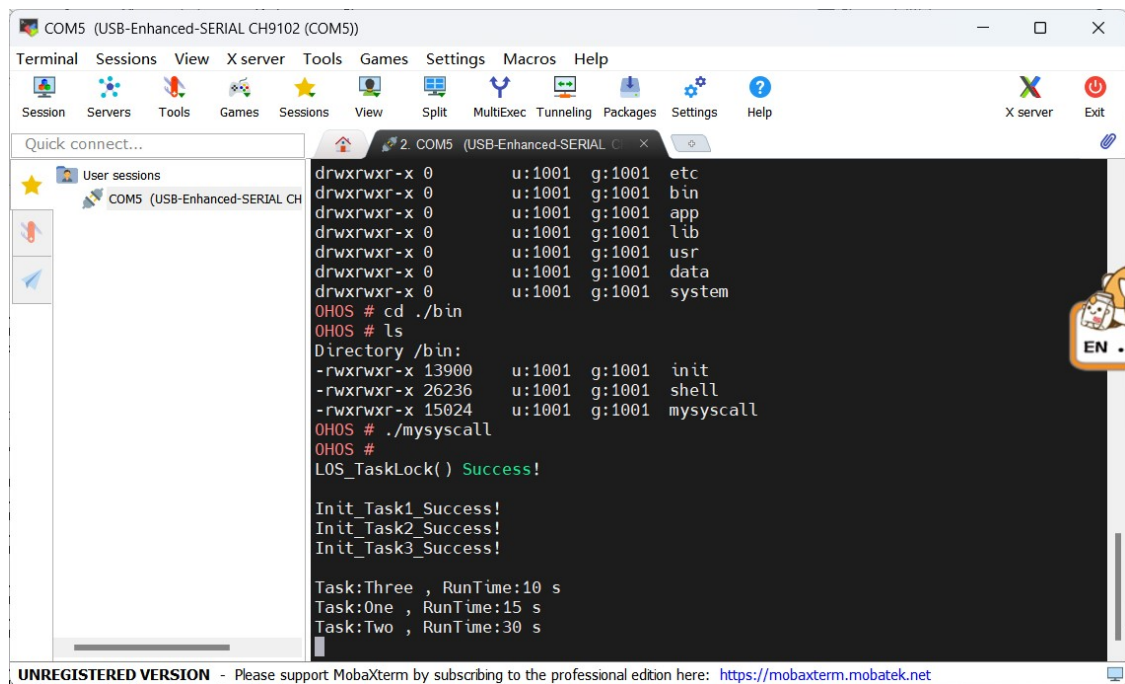
```
initParam.pfnTaskEntry = (TSK_ENTRY_FUNC)PRIOR_THIRD_TASK;
initParam.usTaskPrio = PRIOR;
initParam.pcName = "PRIOR_THIRD_TASK";
initParam.uwStackSize = OS_TASK_RESOURCE_STATCI_SIZE;
initParam.RunTime = 10;
initParam.uwResved = LOS_TASK_STATUS_DETACHED;
ret = LOS_TaskCreate(&THIRD,&initParam);
if(ret != LOS_OK)
{
    LOS_TaskUnlock();
    PRINTK("Failed_3!\r\n");
    return;
}
PRINTK("Init_Task3_Success!\r\n\n");

LOS_TaskUnlock();
return;
}
```

6. 开发板测试

测试结果如下文

4 实验结果



```
COM5 (USB-Enhanced-SERIAL CH9102 (COM5))
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
User sessions
COM5 (USB-Enhanced-SERIAL CH
drwxrwxr-x 0      u:1001 g:1001 etc
drwxrwxr-x 0      u:1001 g:1001 bin
drwxrwxr-x 0      u:1001 g:1001 app
drwxrwxr-x 0      u:1001 g:1001 lib
drwxrwxr-x 0      u:1001 g:1001 usr
drwxrwxr-x 0      u:1001 g:1001 data
drwxrwxr-x 0      u:1001 g:1001 system
OHOS # cd ./bin
OHOS # ls
Directory ./bin:
-rwxrwxr-x 13900 u:1001 g:1001 init
-rwxrwxr-x 26236 u:1001 g:1001 shell
-rwxrwxr-x 15024 u:1001 g:1001 mysyscall
OHOS # ./mysyscall
OHOS #
LOS_TaskLock() Success!
Init_Task1_Success!
Init_Task2_Success!
Init_Task3_Success!

Task:Three , RunTime:10 s
Task:One , RunTime:15 s
Task:Two , RunTime:30 s
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
```

5 实验分析

主要实现思路如下：

- 1.在 TCB 中插入一个 RunTime 变量用于记录任务的运行时间

2.修改优先级队列函数使得运行时间较低的任务优先

实验结果:

实验结果中的 2-4 行代表任务的生成顺序

5-8 行为任务的执行顺序

可以看到 10s 运行时间的 Task3 优先运行, 15s 的 Task1 次之, 最后是 30s 的 Task2, 符合运行时间低优先运行

因此实验成功

6 实验总结

通过本次实验, 我深入理解了 LiteOS 的调度器实现和代码结构, 了解了不同的调度策略对同一批任务的处理运行时间的影响以及他们的优劣。该实验不仅加深了我对 LiteOS 调度器的理解, 还使我学到了很多关于系统修改和优化的经验。

7 参考文献

1.[美]William Stallings 著, 陈向群, 陈 渝 等译《操作系统——精髓与原理设计(第八版)》

8 附录

1. OsPriQueueEnqueue 方法

```
VOID OsPriQueueEnqueue(LOS_DL_LIST *priQueueList, UINT32 *bitMap,
LOS_DL_LIST *priqueueItem, UINT32 priority)
{
    if (LOS_ListEmpty(&priQueueList[priority])) {
        *bitMap |= PRIQUEUE_PRIOR0_BIT >> priority;
        LOS_ListTailInsert(&priQueueList[priority], priqueueItem);
    }
}
```

```

}else{

    LOS_DL_LIST *currentItem = priQueueList[priority].pstNext;

    LOS_DL_LIST *prevItem = &priQueueList[priority];

    while(currentItem != &priQueueList[priority]){

        UINT32 thisTime = *(UINT32*)(priqueueItem + 1);

        UINT32 frontTime = *(UINT32*)(currentItem + 1);

        if(thisTime < frontTime){

            priqueueItem->pstNext = currentItem;

            prevItem->pstNext = priqueueItem;

            if(prevItem->pstPrev){

                priqueueItem->pstPrev = prevItem;

            }

            if(currentItem->pstPrev){

                currentItem->pstPrev = priqueueItem;

            }

            return;

        }

        prevItem = currentItem;

        currentItem = currentItem->pstNext;

    }

    prevItem->pstNext = priqueueItem;

    priqueueItem->pstNext = &priQueueList[priority];

    priqueueItem->pstPrev = prevItem;

}

```

```
}
```

2. 验证测试程序

```
#include "los_printf.h"

#include "los_task_pri.h"

#include "los_base_pri.h"

#include "los_priqueue_pri.h"

#include "los_sem_pri.h"

#include "los_event_pri.h"

#include "los_mux_pri.h"

#include "los_hw_pri.h"

#include "los_exc.h"

#include "los_memstat_pri.h"

#include "los_mp.h"

#include "los_spinlock.h"

#include "los_percpu_pri.h"

#include "los_process_pri.h"


#ifdef __cplusplus

if __cplusplus


#endif


#endif


#include "stdio.h"
```

```
#include "los_task_pri.h"
```

```
UINT32 FIRST;
```

```
UINT32 SECOND;
```

```
UINT32 THIRD;
```

```
#define PRIOR 1
```

```
void PRIOR_FIRST_TASK(VOID)
```

```
{  
  
    PRINTK("Task:One , RunTime:15 s\r\n");  
  
    return;  
}
```

```
void PRIOR_SECOND_TASK(VOID)
```

```
{  
  
    PRINTK("Task:Two , RunTime:30 s\r\n");  
  
    return;  
}
```

```
void PRIOR_THIRD_TASK(VOID)
```

```
{  
  
    PRINTK("Task:Three , RunTime:10 s\r\n");  
  
    return;  
}
```

```

void MySyscall(int num){

    UINT32 ret;

    TSK_INIT_PARAM_S initParam;

    LOS_TaskLock();

    PRINTK("\nLOS_TaskLock() Success!\r\n\n");


    initParam.pfnTaskEntry = (TSK_ENTRY_FUNC)PRIOR_FIRST_TASK;

    initParam.usTaskPrio = PRIOR;

    initParam.pcName = "PRIOR_FIRST_TASK";

    initParam.uwStackSize = OS_TASK_RESOURCE_STATCI_SIZE;

    initParam.RunTime = 15;

    initParam.uwResved = LOS_TASK_STATUS_DETACHED;

    ret = LOS_TaskCreate(&FIRST,&initParam);

    if(ret != LOS_OK)
    {
        LOS_TaskUnlock();

        PRINTK("Failed_1!\r\n");

        return;
    }

    PRINTK("Init_Task1_Success!\r\n");


    initParam.pfnTaskEntry = (TSK_ENTRY_FUNC)PRIOR_SECOND_TASK;

    initParam.usTaskPrio = PRIOR;

```

```

initParam.pcName = "PRIOR_SECOND_TASK";

initParam.uwStackSize = OS_TASK_RESOURCE_STATCI_SIZE;

initParam.RunTime = 30;

initParam.uwResved = LOS_TASK_STATUS_DETACHED;

ret = LOS_TaskCreate(&SECOND,&initParam);

if(ret != LOS_OK)
{
    LOS_TaskUnlock();

    PRINTK("Failed_2!\r\n");

    return;
}

PRINTK("Init_Task2_Success!\r\n");

```

```

initParam.pfnTaskEntry = (TSK_ENTRY_FUNC)PRIOR_THIRD_TASK;

initParam.usTaskPrio = PRIOR;

initParam.pcName = "PRIOR_THIRD_TASK";

initParam.uwStackSize = OS_TASK_RESOURCE_STATCI_SIZE;

initParam.RunTime = 10;

initParam.uwResved = LOS_TASK_STATUS_DETACHED;

ret = LOS_TaskCreate(&THIRD,&initParam);

if(ret != LOS_OK)
{
    LOS_TaskUnlock();

```



```
PRINTK("Failed_3!\r\n");
```

```
return;
```

```
}
```

```
PRINTK("Init_Task3_Success!\r\n\n");
```

```
LOS_TaskUnlock();
```

```
return;
```

```
}
```