

# 实 验 指 南

|  |  |
| --- | --- |
| 课 程： | 嵌入式系统协同设计 |
| 实验序号： | 第2次实验 |
| 实验名称： | LiteOS-M的进程/线程管理创建实验 |
| 院 系： | 软件学院 |
| 专业班级： | 软件工程2102班 |
| 学 号： | U202117254 |
| 姓 名： | 陈德霆 |

2023年07月09日

**一、实验目的**

（1）掌握LiteOS-M创建进程/线程的流程和主要数据结构。

（2）了解进程/线程调度策略对多个线程的并发运行效率的影响。

（1）掌握LiteOS-M内存管理的基本原理和主要数据结构。

（2）了解进程/线程在运行过程对内存的申请和释放流程。

**二、实验内容**

（1）创建3个特定的线程，线程函数的名字分别设定为“DebugThreadHigh”，“DebugThreadMiddle”，“DebugThreadLow”，且优先级有明显差别。三个线程在用户态是死循环，分别输出”H”,”M”,”L”三个字符。跟踪内核创建这些线程的流程，并在相应的内核函数中打印出相应的提示信息，打印出线程数据结构的重要成员变量的取值（突出打印出与调度有关的成员变量），在三个线程被切入和切出的时候分别输出提示信息。打印三个线程在应用态输出的”H”,”M”,”L”三个字符的数量多寡。

**三、实验环境**

宿主机操作系统：Ubuntu 20.04LTS

编译工具：gcc/g++

编辑工具：vscode

**四、实验设计思路**

**4.1 华为LiteOS-M内核目录和源文件结构**

\DebugThreadLow.cpp

\DebugThreadMiddle.cpp

\DebugThreadHigh.cpp

\BUILD.gn

**4.2 LiteOS-M进程/线程管理（或内存管理）的基本原理**

1.进程/线程概念：在 LiteOS-M 中，操作系统主要使用任务（Task）的概念，这与传统操作系统中的线程或进程类似。在这个系统中，每个任务都有其自己的上下文和优先级，并按照优先级进行调度。

2.基本原理：LiteOS-M 是一个抢占式的实时操作系统。这意味着它会在满足一定条件时（例如一个任务的执行时间用尽，或者更高优先级的任务变为可运行状态时），中断当前运行的任务，切换到另一个任务。这种切换是由操作系统的调度器控制的。

3.主要数据结构:LiteOS-M 中最关键的数据结构是任务控制块（ ProcessCB）,PCB 包含了一个任务的所有信息，包括任务的状态（就绪、运行、阻塞等）、优先级、栈指针等。

4.主要函数：LiteOS-M 提供了一系列函数用于任务管理，如 osThreadNew 用于创建新的任务，osDelay 用于让当前任务延迟一段时间，osThreadTerminate 用于结束一个任务。

5.调度机制：LiteOS-M 使用基于优先级的抢占式调度算法。每个任务都有一个优先级，调度器总是选择就绪队列中优先级最高的任务运行。

**4.4 程序结构**

**4.4.1 程序的基本结构**

1.线程函数

DebugThreadxxx是一个线程函数，它无限循环的打印 count2 的值，并调用 osDelay 函数使线程延迟1毫秒，然后开始下一次循环。

1. 线程创建函数

DebugThreadHighDemo中定义了osThreadAttr\_t 结构体 attr 来设置线程的属性，主要包括线程名和优先级的设置。它调用 osThreadNew 函数来创建一个新的线程，线程的入口函数为 DebugThreadxxx。

**4.4.2 关键函数或参数或某种机制**

（1）APP\_FEATURE\_INIT(func);

#define APP\_FEATURE\_INIT(func) \

LAYER\_INITCALL\_DEF(func, app\_feature, "app.feature")

这是一个宏,它将给定的 func 函数与一个特定的初始化层级关联起来，该层级由宏的名字和描述来表示。当系统启动时，这些函数将会按照其关联的层级和优先级顺序执行。

1. osThreadId\_t osThreadNew \

(osThreadFunc\_t func, void \*argument, const osThreadAttr\_t \*attr)

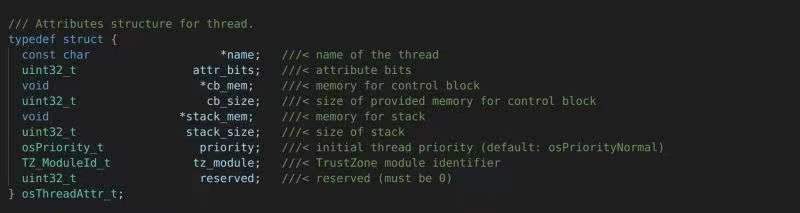
负责创建任务的函数,有三个入口参数func, argument, attr

· func为xTaskCreate()和xTaskCreateStatic()中的第一个入口参数，是一个函数指针，指向执行任务的函数。

· argument为xTaskCreate()和xTaskCreateStatic()中的第四个入口参数，是传递给任务的参数，不用时设为NULL。

· attr 是个Handle句柄，包含了.name，.stack\_size，.priority等属性。

osThreadAttr\_t结构体如下所示：



**五、关键代码分析**

**5.1 程序关键片段一：线程任务函数**

while (1)

{

count2 ++;

int i = 0;

float j = 1;

for(i = 0; i < 100000-count2; i ++)

for(i = 0; i < 300000-count2; i ++)

j+= i;

//打印执行次数

printf("H. Count: %lld.\n",count2,j);

//阻塞函数，单位ms，注意阻塞时间的设置，时间太长体现不出线程之间的优先级

osDelay(1);

}

**5.2 程序关键片段二：线程创建函数**

osThreadAttr\_t attr={0};

printf("[DebugThreadHigh] create DebugThreadHigh!\n");

//设置osThreadAttr\_t结构体

attr.name = "DebugThreadHigh";

attr.stack\_size = 10240;

//设置静态优先级

attr.priority = osPriorityAboveNormal;

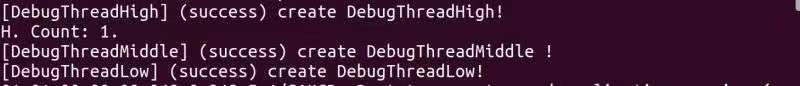
//创建新线程，传入三个参数

if (osThreadNew((osThreadFunc\_t)DebugThreadHigh, NULL, &attr) == NULL) {

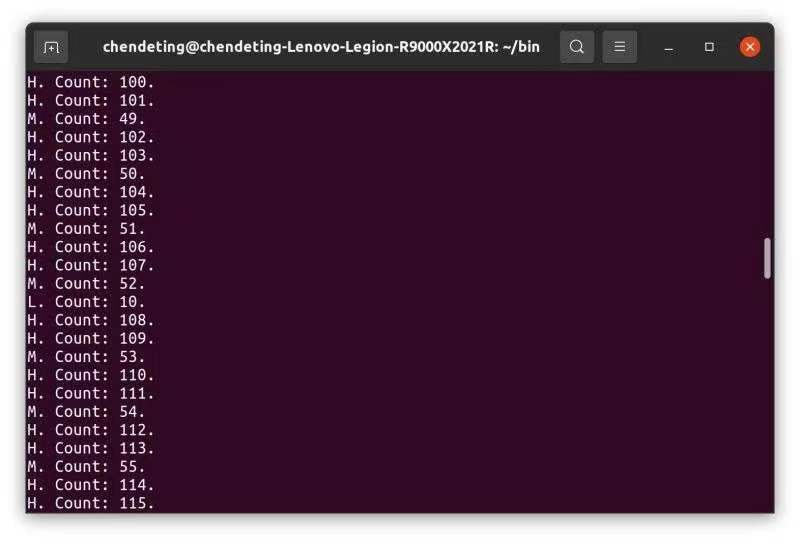
printf("[DebugThreadHigh] Falied to create DebugThreadHigh!\n");

}

1. **程序运行结果和分析**

****

线程创建成功

****

可以看到，不同优先级的线程打印的次数也不同，其中，被设置了高优先级的线程打印的次数最多，中优先级次之，低优先级最少。

**七、实验错误排查和解决方法**

**7.1** Q：线程优先级区别未体现

A：可能是osDealy函数取的阻塞时间不合理，阻塞时间过长（例如>=10)，可能优先级调度不明显，设置为1ms，问题解决。

**7.2 Q：线程创建失败**

A：可能是osThreadNew函数参数选取有问题，attr要被正确设置，attr.priority 取值要合理，实测优先级取osPriorityNormal附近没问题，osPriorityHigh可能会出现线程创建失败的情况。

1. **实验参考资料和网址**
2. 教学课件
3. RTX筆記2 - thread 管理

（3）网上链接1：https://www.bookstack.cn/read/openharmony-1.0-zh-cn/api-api-LinkIoT-OHOS-Init.md