1. * 1. Huawei-LiteOS
        1. 简介

Huawei-LiteOS是轻量级的实时操作系统（典型的配置为64KB的数据内存和96KB的程序闪存），遵循BSD-3**开源**许可协议，能大幅降低设备布置及维护成本，有效降低开发门槛、缩短开发周期[1]。

|  |
| --- |
| **扩展阅读**  开源操作系统的历史是和GNU紧密联系在一起的。从1983年开始的GNU计划：致力于开发一个自由并且完整的类Unix操作系统（包括软件开发工具和各种应用程序），到1991年 Linux内核发布的时候，GNU已经几乎完成了除了系统内核之外的各种必备软件的开发。在Linus Torvalds和其他开发人员的努力下，GNU组件可以运行于Linux内核之上，整个内核是基于 GNU 通用公共许可，也就是GPL（GNU General Public License，GNU通用公共许可证）的，但是Linux内核并不是GNU 计划的一部分。1994年3月，Linux1.0版正式发布，Marc Ewing成立了 Red Hat软件公司，成为最著名的 Linux 分销商之一。 |

2015年，华为在HNC网络大会上，正式推出了“1+2+1”物联网战略，即“一个轻量级物联网操作系统，两种接入方式，一个物联网平台”。Huawei-LiteOS就应运而生了。

Huawei-LiteOS是华为面向IoT领域构建的“统一物联网操作系统和中间件软件平台”，依靠轻量级（内核小于10k）、低功耗（1节5号电池最多可以工作5年）、快速启动、互联互通、安全等关键能力，为开发者提供 “一站式”的完整软件平台，有效降低了开发门槛，缩短了开发周期。它可以广泛应用于智能家居、穿戴式、车联网、制造业等领域，使得物联网开发变得更简单、终端互联更加容易、业务逻辑更加智能、用户体验更加顺畅、数据传输更加安全。

Huawei-LiteOS的内核由四部分组成：软件开发工具包（SDK）、基础内核、设备驱动层和硬件层。其中的SDK部分又包括了端云互通组件、FOTA、JS引擎和传感框架等内容。具体如图3-1所示。



图3-1 Huawei-LiteOS的内核示意图

其中，端云解决方案支持华为的IoT联接管理平台OceanConnect。OceanConnect 是华为在“1+2+1”的物联网战略中提出的，基于物联网、云计算和大数据等技术打造的开放生态环境[2]。

**硬件支持列表：**Huawei-LiteOSLiteOS目前已经适配了30多种开发板，其中包括ST、NXP、GD、MIDMOTION、SILICON、ATMEL等主流厂商的开发板。表3-1列举了几款Huawei-LiteOS支持的开发板：

表3-1 部分Huawei-LiteOS支持的开发板

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **厂商** | **开发板** | **MCU** | **射频芯片** | **传感模块** |
| ST | STM 32F429ZI-NUCLEO | STM 32F429ZI | Integrated | - |
| GD | GD32F450I-EVAL | GD32F450I | Integrated | - |
| Atmel | ARDUINO M0 PRO | ATSAMD21G18 | Integrated | - |
| NXP | LPC54110 | LPC54114J256BD64 | Integrated | 温度传感器、数字麦克风 |
| MIDMOTION | MM32F103 | MM32F103RBT6 | Integrated | 温度传感器 |
| Silicon Labs | EFM32GG-STK3700 | EFM32GG990F1024 | Integrated | 环境光传感器、感应电容式金属传感器 |
| Nuvoton | NuTiny-NUC472H | NUC472H | Integrated | - |
| Nordic Semiconductors | NRF52840-PDK | NRF52840 | Integrated | - |

* + - 1. 关键特性

**(1)基于事件和线程的编程模型：**Huawei-LiteOS的编程模型是基于C语言的，同时支持**事件**和**线程**，开发者只要懂得C语言的基本语法就可以基于Huawei-LiteOS进行物联网应用的开发，上手非常简单，只需要在芯片上适配好Huawei-LiteOS代码包，之后基于Huawei-LiteOS代码包使用C语言开发新的模块业务即可。

**①事件：**Huawei-LiteOS支持基于事件的编程。多任务环境下，任务之间往往需要同步操作，事件是一种实现任务间通信的机制，可用于实现任务间的同步。一个任务可以等待一个或多个事件的发生，可以在任意一个事件发生时唤醒某个任务进行事件处理，也可以在几个事件都发生后才唤醒该任务进行事件处理。

**②线程：**Huawei-LiteOS支持基于线程的编程，可以在线程之间进行通信和切换。线程是竞争系统资源的最小运行单元，线程可以使用或等待CPU、使用内存空间等系统资源，并独立于其它线程运行。在Huawei-LiteOS中，每个线程由线程ID、线程优先级、线程入口函数、线程控制块TCB、线程栈和线程上下文组成。每个线程都有多种运行状态：就绪态、运行态、阻塞态和退出态，这些状态之间的转换如图3-2所示：



图3-2 线程状态迁移示意图

下面将具体说明各状态之间的转换：

* 就绪态->运行态：操作系统在进行线程调度时，将就绪队列中最高优先级的线程调度至处理器上执行，被调度的线程就由就绪态进入了运行态。
* 运行态->阻塞态：正在处理器上执行的线程由于挂起、延时、读信号量等待等原因发生阻塞时，就由运行态进入了阻塞态。
* 阻塞态->就绪态：阻塞的线程由于任务恢复、延时时间超时、读信号量超时或读到信号量等原因被恢复时，就由阻塞态进入了就绪态。
* 就绪态->阻塞态：线程在就绪状态的时候被挂起，就由就绪态进入了阻塞态。
* 运行态->就绪态：在优先级较高的线程被创建或恢复后，操作系统会进行线程调度，将就绪队列中优先级最高的线程调度至处理器上执行，原先在处理器上执行的线程则由运行态进入了就绪态。
* 运行态->退出态：线程执行结束后，其状态就由运行态进入了退出态。
* 阻塞态->退出态：被阻塞的线程调用删除接口，其状态就由阻塞态进入了退出态。

**(2)时间片轮转调度和抢占式调度：**Huawei-LiteOS支持时间片轮转调度和抢占式调度，并且可以在两种调度方式之间进行切换。

**①时间片轮转调度：**时间片轮转调度是指每个线程被分配一个时间片，即该线程允许运行的时间。如果在时间片结束时线程还在运行，处理器将被剥夺并分配给另一个线程。如果线程在时间片结束前线程阻塞或结束，处理器将立即切换给其他线程。

**②抢占式调度：**抢占式调度是指在现行线程运行过程中，如果就绪列表中有更重要或紧迫的线程要运行，则当前运行线程会被迫放弃处理器，系统随即将处理器分配给更重要的线程。

下面的示例将介绍Huawei-LiteOS的实时抢占调度：低优先级线程持续运行，会被高优先级线程打断，高优先级线程直到调用 LOS\_TaskDelay 后，才将 CPU 释放出来。

使用LOS\_TaskCreate创建两个线程，线程优先级分别为4和5：

创建Example05\_Entry入口函数，在Example05\_Entry中先创建优先级为5的高优先级线程，然后创建优先级为4的低优先级线程。

1 T32 Example05\_Entry(VOID) {

2 UINT32 uwRet = LOS\_OK;

3 TSK\_INIT\_PARAM\_S stInitParam = {0};

4 printf("Example05\_Entry\r\n");

5  //创建高优先级线程

6 stInitParam.pfnTaskEntry = Example\_TaskHi;

7 stInitParam.usTaskPrio = TASK\_PRIO\_HI;

8 stInitParam.pcName = "TaskHi";

9 stInitParam.uwStackSize = TASK\_STK\_SIZE;

10 stInitParam.uwArg = (UINT32)pcTextForTaskHi;

11 uwRet = LOS\_TaskCreate(&s\_uwTskHiID, &stInitParam);

12 if (uwRet != LOS\_OK) {

13 printf("Example\_TaskHi create Failed!\r\n");

14 return LOS\_NOK;

15 }

16 //创建高优先级线程

17 stInitParam.pfnTaskEntry = Example\_TaskLo;

18 stInitParam.usTaskPrio = TASK\_PRIO\_LO;

19 stInitParam.pcName = "TaskLo";

20 stInitParam.uwStackSize = TASK\_STK\_SIZE;

21 stInitParam.uwArg = (UINT32)pcTextForTaskLo;

22 uwRet = LOS\_TaskCreate(&s\_uwTskLoID, &stInitParam);

23 if (uwRet != LOS\_OK) {

24 printf("Example\_TaskLo create Failed!\r\n");

25 return LOS\_NOK;

26 }

27 return uwRet;

28 }

代码3-1 LOS\_TaskCreate入口函数

高优先级线程打印回显信息：

|  |
| --- |
| 1 //高优先级线程  2 static VOID \* Example\_TaskHi(UINT32 uwArg) {  3 for (;;) {  4 printf("%s\r\n", (const CHAR \*)uwArg);  5 LOS\_TaskDelay(2000);  6 }  7 } |

代码3-2 高优先级线程

低优先级线程打印回显信息：

|  |
| --- |
| 1 //低优先级线程  2 static VOID \* Example\_TaskLo(UINT32 uwArg) {  3 UINT32 i;  4 for (;;) {  5 printf("%s\r\n", (const CHAR \*)uwArg);  6 for (i = 0; i < TASK\_LOOP\_COUNT; i++) {  7 //占用CPU耗时运行，过程中会被高优先级线程打断  8 }  9 }  10 } |

代码3-3 低优先级线程

运行结果：低优先级线程在执行过程中会被高优先级线程打断。

|  |
| --- |
| Example05\_Entry //进入主程序  Task\_Low is running //低优先级线程执行  Task\_Low is running  Task\_Low is running  Task\_Hi is running //高优先级线程打断  Task\_Low is running //高优先级线程调用LOS\_TaskDelay  Task\_Low is running //低优先级线程重获CPU  Task\_Low is running  Task\_Hi is running //高优先级线程打断 |

代码3-4 线程调度运行结果

**(3)多网络协议栈的支持：**Huawei-LiteOS提供轻量级内核和连接引擎，支持大规模的网络协议，并允许各种类型的连接。如图3-3所示，Huawei-LiteOS支持的短距离网络协议包括BT、Wi-Fi、6LoWPAN、ZigBee等，支持的长距离网络协议包括LTE、NB-IoT等。



图3-3 Huawei-LiteOS的网络协议栈

**①BT协议：**BT是BitTorrent的简称，中文全称“比特流”，是互联网上最热门的一种P2P传输协议。

**②Wi-Fi协议：**Wi-Fi是一种无线局域网技术，使用的频段是2.4G UHF和5G SHF ISM，以微波、激光、红外等无线载波作为传输介质，主要用于解决局域网中终端设备的无线接入。

**③6LoWPAN协议：**6LoWPAN是一种基于IPv6的短距离、低速率、低功耗的无线个域网协议。**Huawei-LiteOS使用6LoWPAN对BT和802.15.4协议进行了封装，使得BLE、802.15.4、Wi-Fi协议在IP层可以互通。**

**④ZigBee协议：**ZigBee是短距离、低功耗的局域网协议，它是基于IEEE 802.15.4标准的，使用的频段是2.4GHz，基本传输速率是250kb/s。

**⑤LTE协议：**LTE是移动设备和数据终端之间高速无线通信的协议。

**⑥NB-IoT协议：**NB-IoT是基于蜂窝的低功率广域网协议，用于实现各种蜂窝设备和服务，只占用约180kHz的带宽。

Huawei-LiteOS的网络协议栈更加注重于**遵循现有的标准**和**重用开源组件**，实现了不同网络协议之间的互通。

|  |
| --- |
| **扩展阅读**  国外也有一个开源操作系统叫做LiteOS，它是由美国伊利诺伊大学于2007年推出的，是一个类UNIX的无线传感网操作系统。不知道是什么原因导致华为要选用相同的名字，在LiteOS的官网也提到了这一点，指出了不确定华为使用同样的名字是否表示Huawei-LiteOS是基于LiteOS设计的或者重用了LiteOS的代码。 |

## 参考文献

Huawei-LiteOS official website. https://developer.huawei.com/ict/cn/site-iot-liteos.

OceanConnect official website. https://developer.huawei.com/ict/cn/site-oceanconnect-next.