第 11 章

C++ 特性支持

C++编程语言提供了语言级别的面向对象支持，在大型系统中使用面向对象的设计方法会让系统结构更为自然化，也容易便于理解。RT-Thread 系统中提供了基本的 C++编程语言支持（GNU GCC 环境下）。

11.1 RT-Thread 使用 C++注意事项

RT-Thread 的编程是一种单一地址空间的编程，同时大多数情况下用于嵌入式系统中，在这种环境下 C++编程需要注意一些事项：

- 禁止使用 RTTI 特性。为了支持 RTTI 特性，C++编译生成的代码中将会添加更多的辅助代码用于表示对象特性，这部分特性在 RT-Thread 的 C++支持中删减掉了。
- 请慎重使用 C++语言中的模板特性，大量使用可能造成代码的急剧膨胀。
- 在 C++全局对象中，请慎重使用操作系统提供的服务，因为在全局对象构造时，操作系统还未初始化。

当使用 RT-Thread 系统提供的其他服务时，可以依照原来的 C 编程模式进行访问调用及封装。

11.2 RT-Thread 移植 C++注意事项

单独的 RT-Thread 移植并不能很好的支持 C++特性（缺少全局对象构造功能），需要在操作系统最后的链接过程中添加部分链接脚本内容，此外也需要在系统初始化时显示的调用全局对象构造部分代码。

以 lumit4510，GNU GCC 版本为例说明 RT-Thread 移植中关于 C++部分的代码。

代码 11 - 1lumit4510_ram.lds 文件中关于 C++构造/析构部分链接脚本

```
. = ALIGN(4);
.ctors :    /* 存放构造部分代码 */
{
    PROVIDE(__ctors_start__ = .);
    KEEP(*(SORT(.ctors.*)))
    KEEP(*(.ctors))
    PROVIDE(__ctors_end__ = .);
```

```
    }

    .dtors :    /* 存放析构部分代码，RT-Thread中未用到 */
    {
        PROVIDE(__dtors_start__ = .);
        KEEP(*(SORT(.dtors.*)))
        KEEP(*(.dtors))
        PROVIDE(__dtors_end__ = .);
    }
```

代码 11 - 2 start.S 启动汇编中关于 C++对象构造部分代码

```
    ; 在跳转到第一个C函数之前的C++全局对象构造部分代码
    ldr     r0, =__ctors_start__     ; 在链接脚本中给出
    ldr     r1, =__ctors_end__       ; 在链接脚本中给出
ctor_loop:
    cmp     r0, r1
    beq     ctor_end                 ; 如果R0 = R1，结束
    ldr     r2, [r0], #4             ; 载入__ctors_start__部分内容
    stmfd   sp!, {r0-r1}             ; R0，R1寄存器压栈
    mov     lr, pc                   ; 保存PC到LR
    bx      r2                       ; 跳转到R2中
    ldmfd   sp!, {r0-r1}             ; 恢复R0 和 R1寄存器
    b       ctor_loop
ctor_end:
```