内存管理之 rt_realloc 和 rt_free

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0

日期：2012/8/13

修订记录

日期	作者	修订历史
2012/8/13	bloom5	创建文档

实验目的

- 快速熟悉了解动态内存的 API rt_realloc。

硬件说明

本实验使用 RT-Thread 官方的 Realtouch 开发板作为实验平台。涉及到的硬件主要为

- 串口 3，作为 rt_kprintf 输出，需要连接 JTAG 扩展板

具体请参见《Realtouch 开发板使用手册》

实验原理及程序结构

实验设计

rt_realloc 用于需要重新分配内存的应用场景。当原先通过 rt_malloc 申请的内存块因为太小而不能满足应用需求时，可以使用 rt_realloc 进行重新分配。内存管理器会查询原有内存块后是否有足够的空闲内存来分配，如果是，则直接将原有内存块空间扩大，返还给申请者；否则从内存堆中重新找一块新的空间返还给申请者，并将原有内存块空间释放。

源程序说明

本实验对应 1_kernel_mem_realloc

系统依赖

在 rtconfig.h 中需要开启

- #define RT_USING_HEAP

此项可选，开启此项可以创建动态线程和动态信号量，如果使用静态线程和静态信号量，则此项不是必要的

- #define RT_USING_CONSOLE

此项必须，本实验使用 rt_kprintf 向串口打印按键信息，因此需要开启此项

主程序说明

我们在这里只创建一个静态线程来演示动态内存的分配，重新分配和释放。

线程 TCB 和栈

```
struct rt_thread thread1;
char thread1_stack[512];
```

线程入口程序

```
void thread1_entry(void* parameter)
{
    int i,j = 1;
    char *ptr[10]; /* 用于放置 10 个分配内存块的指针*/
    /* 对指针清零*/
    for (i = 0; i <10; i ++)
    ptr[i] = RT_NULL;
    while(j--)
    {
        for (i = 0; i <10; i++)
       {
           /* 每次分配(1 <<i)大小字节数的内存空间*/
           ptr[i] = rt_malloc(10);
           /* 如果分配成功*/

           if (ptr[i] != RT_NULL)
           {
               rt_kprintf("get memory: 0x%x\n", ptr[i]);
               rt_realloc(ptr[i],16);
               /* 如果分配成功*/
               if (ptr[i] != RT_NULL)
               {
                   rt_kprintf("memory realloc success!\n");
                   /* 释放内存块*/
                   rt_free(ptr[i]);
                   rt_kprintf("memory free success!\n");
                   ptr[i] = RT_NULL;
               }
            }
        }
    }
}
```

编译调试及观察输出信息

编译请参见《RT-Thread 配置开发环境指南》完成编译烧录，参考《Realtouch 开发板使用手册》完成硬件连接，连接扩展板上的串口和 jlink。

运行后可以看到如下信息：

```
 \ | /
- RT -     Thread Operating System
 / | \     1.1.0 build Aug 13 2012
 2006 - 2012 Copyright by rt-thread team
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
get memory: 0x20000c58
memory realloc success!
memory free success!
```

结果分析

本例程主要想要体现的就是内存动态的重新分配及释放，内存分配的操作可以参见上一节内容，几乎完全相同。

使用 for 循环依次对每个内存块分配空间，并将分配内存块的首地址保存到 ptr[i]中，将首地址打印出来，然后再重新分配，如果成功打印成功的消息，然后将再将内存释放。

```
ptr[i] = rt_malloc(10);
          /* 如果分配成功*/
if (ptr[i] != RT_NULL)
   {
      rt_kprintf("get memory: 0x%x\n",  ptr[i]);
      rt_realloc(ptr[i],16);
      /* 如果分配成功*/
      if (ptr[i] != RT_NULL)
      {
         rt_kprintf("memory realloc success!\n");
         /* 释放内存块*/
         rt_free(ptr[i]);
         rt_kprintf("memory free success!\n");
         ptr[i] = RT_NULL;
      }

   }
```