内存管理之 rt_malloc 和 rt_free

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0

日期：2012/8/12

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管理相关的 API

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_malloc 和 rt_free 是内存管理中最常用接口，分别用来申请和释放内存。本实验的主要设计目的是帮助读者了解内存管理中动态内存的分配和释放这两个 API 的基本使用方式。

源程序说明

本实验对应 1_kernel_mem_malloc

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和释放。

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

线程入口程序

```
void thread1_entry(void* parameter)
{
    int i,j;
    char *ptr[20]; /* 用于放置 20 个分配内存块的指针*/
    /* 对指针清零*/
    for (i = 0; i < 20; i ++)
        ptr[i] = RT_NULL;
    for(j = 0; j < 2; j ++ )e
    {
        for (i = 0; i <20; i++)
        {
        /* 每次分配(1 <<i)大小字节数的内存空间*/
            ptr[i] = rt_malloc(1 <<i);
            /* 如果分配成功*/
            if (ptr[i] != RT_NULL)
            {
                rt_kprintf("get memory: 0x%x\n",
ptr[i]);
                /* 释放内存块*/
                rt_free(ptr[i]);
                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 10 2012
 2006 - 2012 Copyright by rt-thread team
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
get memory: 0x20000c58
```

结果分析

本例程主要想要体现的就是内存动态的分配及释放，首先定义了一个指针数组，也就是定义了 20 个内存块，指针用来记录内存块的首地址

```
char *ptr[20];
```

将其中每个指针清零，

```
 for (i = 0; i < 20; i ++)
       ptr[i] = RT_NULL;
```

然后使用 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，

```
 for (i = 0; i <20; i++)
       {
       /* 每次分配(1 <<i)大小字节数的内存空间*/
           ptr[i] = rt_malloc(1 <<i);
           /* 如果分配成功*/
           if (ptr[i] != RT_NULL)
           {
               rt_kprintf("get memory: 0x%x\n",
ptr[i]);
               /* 释放内存块*/
               rt_free(ptr[i]);
               ptr[i] = RT_NULL;
           }
       }
```

可以看到分配成功后，将打印所分配内存块的首地址，随机将内存释放，这是因为如果内存块用完后不及时释放，可能会造成内存泄露等一系列问题，相关内容可以查阅 RT-Thread 编程手册。