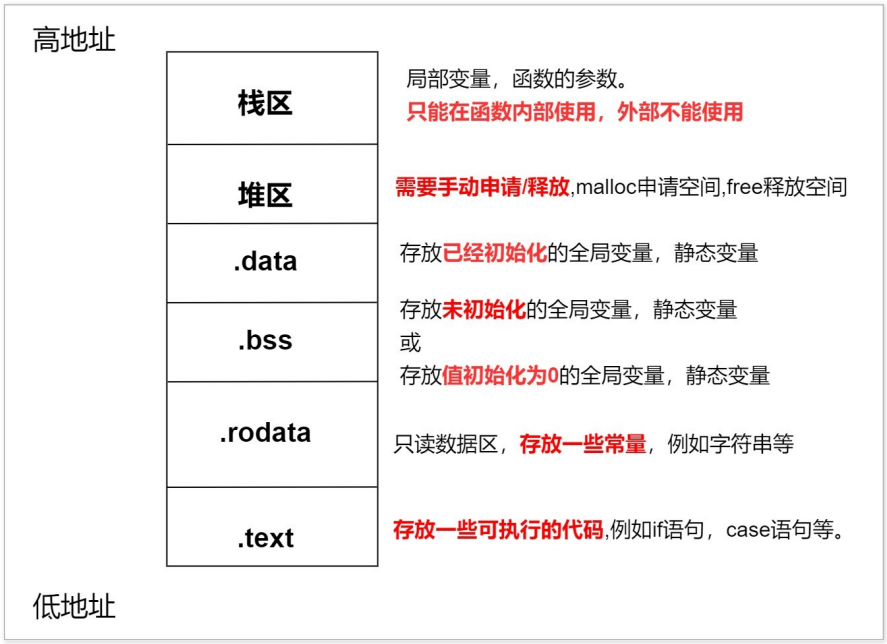
3.3 C 语言堆区内存管理_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.3 C 语言堆区内存管理涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

3.C 语言堆区内存管理

一. C 语言编译的内存分配



二. 堆区空间的分配

1. ### malloc 函数

```
#include <stdlib.h>

void  *malloc(unsigned int size)
功能：从堆区分配内存
参数：
@size    分配内存的字节数
返回值：
成功返回分配内存的首地址，失败返回NULL
```

1. ### free 函数

```
#include <stdlib.h>

void  free(void *ptr)
```

```
功能：释放内存
参数：
@ptr  分配内存的首地址
返回值：
无
```

1. ## memset 函数

```
#include <string.h>

void *memset(void *s, int c, size_t n);
功能：把s所指向内存区域的前n个字节，全部置为c
参数：
@s   想要操作内存区域的首地址
@c   内存区域填充的值
@n   需要填充的字节数
返回值：
成功返回s所指向的地址，
失败返回NULL
```

示例代码 1：

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int *p = NULL;

    p = (int *)malloc(sizeof(int));
    if(NULL == p)
    {
        printf("malloc is fail!\n");
        return -1;
    }
    *p = 800;

    printf("*p = %d\n",*p);

    free(p);
    p = NULL;
    return 0;
}
```

运行结果：

```
*p = 800
```

示例用法 2：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
    int *p = NULL;
    while(1)
    {
        p = (int *)malloc(sizeof(int) * 0xffffff);
        if(NULL == p)
        {
            printf("malloc is fail!\n");
            return -1;
        }
        *p = 800;
        //free(p);
    }
    return 0;
}
```

运行结果：

若是不添加 free§，堆区空间会一直申请，会造成空间耗尽，申请失败。

```
Killed
```

示例用法 3：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define N 5

int *get_memory_addr()
{
        int *addr = NULL;

        addr = (int *)malloc(sizeof(int) * N);
        if(NULL == addr)
        {
                printf("malloc is fail!\n");
                return NULL;
        }
        memset(addr,0,sizeof(sizeof(int) * N));

        return addr;
}

void input_array(int *p)
{
        int i = 0;
        printf("please input %d data : ",N);
        for(i = 0;i < N;i++)
        {
                scanf("%d",&p[i]);
        }
}

void ouput_array(int *p)
{
        int i = 0;
        for(i = 0;i < N;i++)
        {
                printf("%d ",p[i]);
        }
        printf("\n");
}

int main()
{

        int *t = NULL;

        t = get_memory_addr();

        input_array(t);
        ouput_array(t);

        free(t);
        t = NULL;
        return 0;
}
```

运行结果：

```
please input 5 data : 10 20 30 40 50
10 20 30 40 50
```

1.
使用原则

- 需要使用多少内存就分配多少内存，不要过多分配
- 使用完以后一定要 free（）。我们自己申请多少，就要手动释放多少，要不然可能会造成内存泄漏
  【死循环的时候，内存空间没有及时的释放掉，CPU 一直占用内存空间。】

- 当一个程序在操作系统中运行结束后，它运行过程中分配的内存都会被释放掉。

三. 课后任务

练习

```
typedef struct
{
    char name[20];
    int id;
    int score;
}s_t;
```

1. 设计一个s_t *get_memeory_addr()函数,要求在堆区为上述结构体分配空间。
2. 设计一个void input_student(s_t *s)函数，要求用户从键盘输入数据给s中的变量。
3. 设计一个void optput_student(s_t *s)函数，要求输出用户输入的数据。
4. 设计一个main()函数调用上述数据。

- 划线
- 写笔记

学习要认真，笔记应当先

公开笔记 0/1000 提交

Sunny_SunshineX

删除 编辑

全文完

本文由 简悦 SimpRead 优化，用以提升阅读体验

使用了 全新的简悦词法分析引擎 beta，点击查看详细说明