阅读：9,806　　作者：解学武

串的堆分配存储结构（C语言详解版）

上一节　　下一节

串的堆分配存储，其具体实现方式是采用动态数组存储字符串。

通常，编程语言会将程序占有的内存空间分成多个不同的区域，程序包含的数据会被分门别类并存储到对应的区域。拿 C 语言来说，程序会将内存分为 4 个区域，分别为堆区、栈区、数据区和代码区，其中的堆区是本节所关注的。

与其他区域不同，堆区的内存空间需要程序员手动使用 malloc 函数申请，并且在不用后要手动通过 free 函数将其释放。

C 语言中使用 malloc 函数最多的场景是给数组分配空间，这类数组称为动态数组。例如：

```
char * a = (char*)malloc(5*sizeof(char));
```

此行代码创建了一个动态数组 a，通过使用 malloc 申请了 5 个 char 类型大小的堆存储空间。

动态数组相比普通数组（静态数组）的优势是长度可变，换句话说，根据需要动态数组可额外申请更多的堆空间（使用 relloc 函数）：

```
a = (char*)realloc(a, 10*sizeof(char));
```

通过使用这行代码，之前具有 5 个 char 型存储空间的动态数组，其容量扩大为可存储 10 个 char 型数据。

下面给大家举一个完整的示例，以便对串的堆分配存储有更清楚地认识。该程序可实现将两个串（"data.bian" 和 "cheng.net"）合并为一个串：

```
01.  #include <stdio.h>
02.  #include <stdlib.h>
03.  #include <string.h>
04.  int main()
05.  {
06.      char * a1 = NULL;
07.      char * a2 = NULL;
08.      a1 = (char*)malloc(10 * sizeof(char));
09.      strcpy(a1, "data.bian");//将字符串"data.bian"复制给a1
10.      a2 = (char*)malloc(10 * sizeof(char));
11.      strcpy(a2, "cheng.net");
```

```
    int lengthA1 = strlen(a1);//a1串的长度
    int lengthA2 = strlen(a2);//a2串的长度
    //尝试将合并的串存储在 a1 中，如果 a1 空间不够，则用realloc动态申请
    if (lengthA1 < lengthA1 + lengthA2) {
        a1 = (char*)realloc(a1, (lengthA1 + lengthA2+1) * sizeof(char));
    }
    //合并两个串到 a1 中
    for (int i = lengthA1; i < lengthA1 + lengthA2; i++) {
        a1[i] = a2[i - lengthA1];
    }
    //串的末尾要添加 \0，避免出错
    a1[lengthA1 + lengthA2] = '\0';
    printf("%s", a1);
    //用完动态数组要立即释放
    free(a1);
    free(a2);
    return 0;
}
```

程序运行结果：

```
data.biancheng.net
```

注意，程序中给 a1 和 a2 赋值时，使用了 strcpy 复制函数。这里不能直接用 a1 ="data.biancheng"，程序编译会出错，报错信息为 "没有 malloc 的空间不能 free"。因为 strcpy 函数是将字符串复制到申请的存储空间中，而直接赋值是字符串存储在别的内存空间（本身是一个常量，放在数据区）中，更改了指针 a1 和 a2 的指向，也就是说，之前动态申请的存储空间虽然申请了，结果还没用呢就丢了。

上一节 下一节