北京交通大学
BEIJING JIAOTONG UNIVERSITY

字符串和字符数组

——C语言基础知识

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

int main()
{
    char *p = NULL;
    int i;
    p = (char *)malloc(6*sizeof(char));

    p = "abcdefg";
    printf("%s\n", p);
    printf("0x%x\n", p);

    free(p);
    printf("%s\n", p);
    printf("0x%x\n", p);

    return 0;
}
```

运行结果

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
abcdefg
0x405064
0x405064
abcdefg

Process finished with exit code 0
```

问题

- malloc()申请了6个字符的空间，为什么可以存放7个字符？
- 为什么free()执行后，p的地址以及地址中存储的数据都仍然可以访问？

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

int main()
{
    char *p = NULL;
    int i;
    p = (char *)malloc(6*sizeof(char));
    printf("0x%x\n", p);
    p = "abcdefg";
    printf("%s\n", p);
    printf("0x%x\n", p);

    free(p);
    printf("%s\n", p);
    printf("0x%x\n", p);

    return 0;
}
```

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
0x1e15e8
abcdefg
0x40506a
abcdefg
0x40506a

Process finished with exit code 0
```

1。C语言对字符串的约定

字符串借助于字符型一维数组存放，用’\0’作为字符串结束符。

2. 表示字符串常量的约定

C没有字符串数据类型，但允许使用“字符串常量”，即用双引号引起。

3. 字符串给出的是地址值

字符串常量存储时是放入字符数组中，系统为其自动加‘\0’，

```
p = (char *)malloc(6*sizeof(char));
printf("0x%x\n",p);
p = "abcdefg";
```

因此，赋值语句之后，将字符串常量的地址赋给了指针p

那么如何在制定地址存入字符串？

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

int main()
{
    char *p = NULL;
    int i;
    p = (char *)malloc(6*sizeof(char));
    printf("0x%x\n", p);
    strcpy(p, "abcdefg");
    printf("%s\n", p);
    printf("0x%x\n", p);

    free(p);
    printf("%s\n", p);
    printf("0x%x\n", p);

    return 0;
}
```

为什么指针p仍然存储原先的地址，但是无法打印改地址存储的数据？

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
0xb415e8
abcdefg
0xb415e8
??
0xb415e8

Process finished with exit code 0
```

void free(void *ptr)

头文件：malloc.h或stdlib.h

作用：释放malloc(或calloc、realloc)函数给指针变量分配的内存空间的函数

使用后该指针变量一定要重新指向NULL，防止野指针出现，有效 规避误操作。

切记：

- 函数free()不会自动将指针ptr置零！

那么如何在制定地址存入字符串？

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

int main()
{
    char *p = NULL;
    int i;
    p = (char *)malloc(6*sizeof(char));
    printf("0x%x\n", p);
    for(i =0; i<6; i++)
    {
    *p=97+i;
    p++;
    }
    printf("%s\n", p);
    printf("0x%x\n", p);

    free(p);
    printf("%s\n", p);
    printf("0x%x\n", p);

    return 0;
}
```

为什么无法打印出字符串？

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
0xad15e8
?
0xad15ee
?
0xad15ee

Process finished with exit code 0
```

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

int main()
{
    char *p = NULL;
    char *q = NULL;
    int i;
    p = (char *)malloc(6*sizeof(char));
    q=p;
    printf("0x%x\n",p);
    for(i =0; i<6; i++)
    {
    *q=97+i;
    q++;
    }
    printf("%s\n",p);
    printf("0x%x\n",p);

    free(p);
    printf("%s\n",p);
    printf("0x%x\n",p);

    return 0;
}
```

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
0x1215e8
abcdef
0x1215e8
?
0x1215e8

Process finished with exit code 0
```

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

int main()
{
    char *p = NULL;
    int i;
    p = (char *)malloc(6*sizeof(char));

    printf("0x%x\n",p);
    for(i =0; i<6; i++)
    {
        p++;
        *p=97+i;
    }
    printf("%s\n",p);
    printf("0x%x\n",p);

    free(p);
    printf("%s\n",p);
    printf("0x%x\n",p);

    return 0;
}
```

为什么可以打印出指针p所指向的字符？

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
0x7415e8
f
0x7415ee
f
0x7415ee

Process finished with exit code 0
```

void free(void *ptr)

头文件：malloc.h或stdlib.h

作用：释放malloc(或calloc、realloc)函数给指针变量分配的内存空间的函数

使用后该指针变量一定要重新指向NULL，防止野指针出现，有效 规避误操作。

切记：

- 函数free()的参数必须是malloc()的返回值 ！

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

int main()
{
    char *p = NULL;
    int i;
    p = (char *)malloc(6*sizeof(char));

    printf("0x%x\n",p);
    for(i =0; i<6; i++)
    {
        p++;
        *p=97+i;
    }
    printf("%s\n",p);
    printf("0x%x\n",p);

    free(p);
    printf("%s\n",p);
    printf("0x%x\n",p);

    return 0;
}
```

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
0x7415e8
f
0x7415ee
f
0x7415ee

Process finished with exit code 0
```

都是打印p，为什么打印格式不同输出分别为字符和地址？

字符串的输入和输出

- 字符用%c，字符串用%s
- 用scanf函数输入时，输入项或为字符数组名，或为字符数组元素的地址，或为字符指针变量。scanf(“%s”,str);
  - ✓ 输入时白字符作为分割符；输出时不自动换行
- 用printf函数输出时， printf(“%s”,str);

如何让指针指向字符串

- 通过赋初值使指针指向字符串

```
char *ps1="from one";
char str[ ]="from two", *ps2=str;
```

- 通过赋值运算使指针指向字符串

```
例1: char *ps1;
     ps1="from one";
例2: char str[ ]="from two",*ps2;
     ps2=str;
```

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

int main()
{
    char p[] = "abcdef";
    printf("%s\n", p);
    printf("0x%x\n", p);
    return 0;
}
```

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
abcdef
0x60ff29

Process finished with exit code 0
```

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

int main()
{
    char p[10] = {1, 2, 2, 3, 4, 5, 6, 7, 8, 9};
    int i=0;
    for(i =0; i<6; i++)
    {
        p[i]=97+i;
    }
    printf("%s\n",p);
    printf("0x%x\n",p);
    return 0;
}
```

```
D:\CProject\untitled2\cmake-build-debug\untitled2.exe
abcde    ═
0x60ff22

Process finished with exit code 0
```

用数组存储字符串

例如：“ABCD"

```
char str[5]={'A','B','C','D','\0'};
char str[ ]={'A','B','C','D','\0'};
char str[ ]={"ABCD"};
char str[ ]="ABCD";
```

1. 不可用赋值语句对字符数组整体赋值

例1：

```
char mark[10];
```

mark=“C Program”（不可以为数组名重新赋值）

例2:

```
char str1[10]="computer",str2[10];
```

str2=str1;

2. 逐个元素赋值，人为加入串尾标志

例：

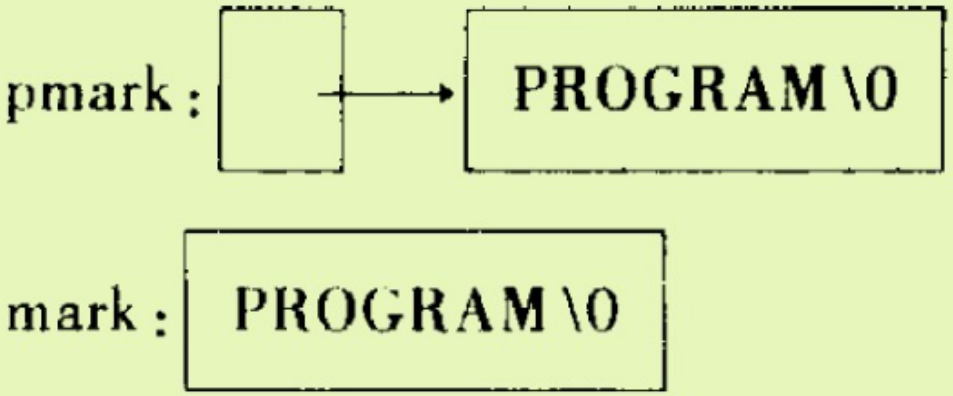
```
char mark[5];
mark[0]='A'; mark[1]='B'; mark[2]='C';
mark[3]='D'; mark[4]='\0';
```

也可以：

```
for(i=0;i<4;i++)
    scanf("%c",&mark[i]);
mark[4]='\0';（scanf函数无法读入 '\0'）
```

字符数组存储字符串和指针指向字符串的区别

```
char mark[ ]="PROGRAM";
char *pmark="PROGRAM";
```



用字符数组保存字符串可靠；用指针指向字符串不可靠，一但指针有了新指向，当前的字符串将“丢失”。