第七章 字符串处理函数

1.1 获取字符串长度函数

头文件：#include <string.h>
函数定义：size_t strlen(const char *s);
函数功能：
　　测字符指针 s 指向的字符串中字符的个数，不包括'\0'
返回值：字符串中字符个数

例 1：

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str1[20]="hello";
    char *str2 ="hello";
    printf("%d\n",sizeof(str1)); //20
    printf("%d\n",sizeof(str2));//4
    printf("%d\n",strlen(str1));//5
    printf("%d\n",strlen(str2));//5
    return 0;
}
```

sizeof 是个关键字，测量数据的占用内存空间大小。
如果测量的是数组的名字，则测的是数组占多少个字节
如果 sizeof 测的是指针变量，则测的是指针变量本身占几个字节，32 平台下结果为 4

strlen 是个库函数，它测的是字符指针指向的字符串中字符的个数，不管指针是数组的名字，还是个指针变量。

1.2 字符串拷贝函数

头文件：#include <string.h>
函数的定义：char *strcpy(char *dest, const char *src);
函数的说明：
拷贝 src 指向的字符串到 dest 指针指向的内存中，'\0'也会拷贝
函数的返回值：
　　目的内存的地址

注意：在使用此函数的时候，必须保证 dest 指向的内存空间足够大，否则会出现内存污染。

char *strncpy(char *dest, const char *src, size_t n);
函数的说明：
将 src 指向的字符串前 n 个字节，拷贝到 dest 指向的内存中
返回值:目的内存的首地址
注意：
1、strncpy 不拷贝 '\0'
2、如果 n 大于 src 指向的字符串中的字符个数，则在 dest 后面填充 n-strlen(src)个 '\0'

例 2：

```
#include<stdio.h>
#include<string.h>
int main()
{
    char buf[100]="aaaaaaaaaaaaaaaaaaaaaaaaaaaa";
    strncpy(buf,"helloworld",5);
    printf("%s\n",buf);
}
```

结果为 helloaaaaaaaaaaaaaaaaaaaaaaa
验证了不拷贝'\0'

例 3：

```
#include<stdio.h>
#include<string.h>
int main()
{
    int len,i;
    char buf[100]="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
    len=strlen(buf);
    strncpy(buf,"helloworld",15);

    for(i=0;i<len;i++)
        printf("%c",buf[i]);

    printf("\n");
}
```

验证了：
如果 n 大于 src 指向的字符串中的字符个数，则在 dest 后面填充 n-strlen(src)个 '\0'

1.3 字符串追加函数

头文件：#include <string.h>
函数定义：char *strcat(char *dest, const char *src);
函数功能：
strcat 函数追加 src 字符串到 dest 指向的字符串的后面。追加的时候会追加'\0'
注意：保证 dest 指向的内存空间足够大。

例 4：

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str[20]="aa\0aaaaaaaaaaaaaaaa";
    char *src ="hello";
    strcat(str,src);
    printf("%s\n",str);
    return 0;
}
```

结果是 aahello
验证了追加字符串的时候追加'\0'

char *strncat(char *dest, const char *src, size_t n);
追加 src 指向的字符串的前 n 个字符，到 dest 指向的字符串的后面。
注意如果 n 大于 src 的字符个数，则只将 src 字符串追加到 dest 指向的字符串的后面
追加的时候会追加'\0'

例 5：

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str[20]="aa\0aaaaaaaaaaaaaaaa";
    char *src ="hello";
    strncat(str,src,3);
    printf("%s\n",str);
    return 0;
}
```

结果为：aahel
验证了会追加'\0'

1.4 字符串比较函数

strcmp/strncmp　　//比较
头文件：#include <string.h>

函数定义:int strcmp(const char *s1, const char *s2);
函数说明：
　　比较 s1 和 s2 指向的字符串的大小，
　　比较的方法：逐个字符去比较 ascII 码，一旦比较出大小返回。
　　如过所有字符都一样，则返回 0

返回值：
　　如果 s1 指向的字符串大于 s2 指向的字符串　返回 1
　　如果 s1 指向的字符串小于 s2 指向的字符串　返回-1
　　如果相等的话返回 0
int strncmp(const char *s1, const char *s2, size_t n);
函数说明：比较 s1 和 s2 指向的字符串中的前 n 个字符

例 6：

```
#include <stdio.h>
#include <string.h>
int main()
{
    char *str1 = "hello world";
    char *str2 = "hello kitty";

    if(   strcmp(str1,str2) == 0)
        printf("str1==str2\n");
    else if(strcmp(str1,str2) > 0)
        printf("str1>str2\n");
    else
        printf("str1<str2\n");

    if(   strncmp(str1,str2,5) == 0)
        printf("str1==str2\n");
    else if(strncmp(str1,str2,5) > 0)
        printf("str1>str2\n");
    else
        printf("str1<str2\n");

    return 0;
```

```
}
```

1.5 字符查找函数

头文件：#include <string.h>

函数定义：char *strchr(const char *s, int c);
函数说明：
　　在字符指针 s 指向的字符串中，找 ascii 码为 c 的字符
注意，是首次匹配，如果过说 s 指向的字符串中有多个 ASCII 为 c 的字符，则找的是第 1 个字符
返回值：
　　找到了返回找到的字符的地址，
　　找不到返回 NULL，

函数定义：char *strrchr(const char *s, int c);
函数的说明：末次匹配
　　在 s 指向的字符串中，找最后一次出现的 ASCII 为 c 的字符，
返回值：
　　末次匹配的字符的地址。

1.6 字符串匹配函数

#include <string.h>
char *strstr(const char *haystack, const char *needle);
函数说明：
　　在 haystack 指向的字符串中查找 needle 指向的字符串，也是首次匹配
返回值：
　　找到了：找到的字符串的首地址
　　每找到：返回 NULL

例 7：

```
#include <string.h>
#include <stdio.h>
int main()
{
    char str1[20]="jfsdjklsd43$#$53jklj4t5";
    char ch='$';
    char str2[20]="$#$";
    char *result;
```

```
    result=strchr(str1,ch);
    printf("%s\n",result);
    printf("%d\n",result-str1);

    result=strstr(str1,str2);
    printf("%s\n",result);
    printf("%d\n",result-str1);
    return 0;
}
```

1.7 字符串转换数值

atoi/atol/atof //字符串转换功能
头文件：#include <stdlib.h>
函数的定义：int atoi(const char *nptr);
函数的功能：
将 nptr 指向的字符串转换成整数，返回

```
例 8：
int num;
num=atoi(“123”);
则 num 的值为 123
```

long atol(const char *nptr);
double atof(const char *nptr);

1.8 字符串切割函数

头文件：#include <string.h>
函数定义：char *strtok(char *str, const char *delim);
函数的功能：
字符串切割，按照 delim 指向的字符串中的字符，切割 str 指向的字符串。
其实就是在 str 指向的字符串中发现了 delim 字符串中的字符，就将其变成’\0’,
调用一次 strtok 只切割一次，切割一次之后，再去切割的时候 strtok 的第一个参数
传 NULL，意思是接着上次切割的位置继续切

注意如果 str 字符串中出现了连续的几个 delim 中的字符，则只将第一个字符变成’\0’

```
例 9：
#include<string.h>
int main()
{
    char str[100]="小明:21,,,.男.女,北京:haidian";
```

```
    char *p=":,.";
    char *q[7];
    int i=0,j;
    q[i]=strtok(str,p);
    while(q[i]!=NULL)
    {
        i++;
        q[i]=strtok(NULL,p);
    }
    for(j=0;j<i;j++)
    {
        printf("q[%d]: %s\n",j,q[j]);
    }
    printf("str=%p\n",str);
    printf("q[0]=%p\n",q[0]);
}
```

1.9 格式化字符串操作函数

int sprintf(char *buf, const char *format, …);

\\输出到 buf 指定的内存区域。

例：

```
char buf[20];
sprintf(buf,"%d:%d:%d",2013,10,1);
printf("buf=%s\n",buf);
```

int sscanf(const char *buf,const char *format, …);

\\从 buf 指定的内存区域中读入信息

例：

```
int a, b, c;
sscanf("2013:10:1", "%d:%d:%d", &a, &b, &c);
printf("%d  %d  %d\n",a,b,c);
```

例 11：

```
#include <stdio.h>
#include <string.h>
int main()
{
    char buf[20];
    int a, b, c;

    sprintf(buf,"%d:%d:%d",2013,10,1);
```

```
    printf("buf=%s\n",buf);//结果为 2013:10:1

    sscanf("2013:10:1", "%d:%d:%d", &a, &b, &c);
    printf("a=%d,b=%d,c=%d\n",a,b,c); //结果为 a=2013,b=10,c=1
    return 0;
}
```

sscanf 高级用法

1、跳过数据：%*s 或%*d

例：sscanf("1234　5678", "%*d %s", buf);

```
例 12：
#include<stdio.h>
int　main()
{
    char buf[20];
    sscanf("1234 5678","%*d %s",buf);//跳过 1234 ,然后隔一个空格获取字符串
    printf("%s\n",buf);
}
```

结果为 5678

2、读指定宽度的数据：%[width]s

例：sscanf("12345678", "%4s", buf);

```
例 13：
#include<stdio.h>
int　main()
{
    char buf[20];
    sscanf("12345678","%4s ",buf);//从字符串中获取字符串，只要 4 个字节，存放在 buf 中
    printf("%s\n",buf);
}
```

3、支持集合操作：只支持获取字符串

%[a-z] 表示匹配 a 到 z 中任意字符(尽可能多的匹配)

```
例 14：
#include<stdio.h>
int　main()
{
    char buf[20];
    sscanf("agcd32DajfDdFF","%[a-z]",buf);//从字符串中获取输入只要’ a’ 和’ z’ 之间的字符，碰到不在范围内的，就终止了
```

```
    printf("%s\n",buf);//结果为 agcd
}
```

%[aBc] 匹配 a、B、c 中一员，贪婪性

%[^aFc] 匹配非 a Fc 的任意字符，贪婪性

%[^a-z]表示读取除 a-z 以外的所有字符

1.10 const：

1:修饰普通变量，代表只读的意思
const int a=100;//定义了一个只读变量 a 值为 100
以后在程序中，不能再给 a 赋值了

a=200;//错误的，a 只读

2：const 修饰指针

(1)、const char *str

意思是 str 指向的内存的内容不能通过 str 来修改

用来保护 str 指向的内存的内容

但是 str 的指向是可以改变的

char * strcpy(char *dest,const char *src);

(2)、char * const str

意思是 str 是只读的变量，str 不能指向别的地方，
但是 str 指向的内存的内容，是有可能可以修改的

(3)、const char * const str

str 不能指向别的地方，指向的内存的内容也不能通过 str 去修改