### （1）memcmp函数

C 库函数 int memcmp(const void \*str1, const void \*str2, size\_t n)) 把存储区 str1 和存储区 str2 的前 n 个字节进行比较。

声明

下面是 memcmp() 函数的声明。

int memcmp(const void \*str1, const void \*str2, size\_t n)

参数

str1 -- 指向内存块的指针。

str2 -- 指向内存块的指针。

n -- 要被比较的字节数。

返回值

如果返回值 < 0，则表示 str1 小于 str2。

如果返回值 > 0，则表示 str2 小于 str1。

如果返回值 = 0，则表示 str1 等于 str2。

### （2）memset函数

库函数：string.h

描述

C 库函数 **void \*memset(void \*str, int c, size\_t n)** 复制字符 **c**（一个无符号字符）到参数 **str** 所指向的字符串的前 **n** 个字符。

声明

下面是 memset() 函数的声明。

void \*memset(void \*str, int c, size\_t n)

参数

**str** -- 指向要填充的内存块。

**c** -- 要被设置的值。该值以 int 形式传递，但是函数在填充内存块时是使用该值的无符号字符形式。

**n** -- 要被设置为该值的字节数。

返回值

该值返回一个指向存储区 str 的指针。

### （3）sprintf函数

功能

把格式化的数据写入某个字符串缓冲区。  
头文件

stdio.h  
原型

int sprintf( char \*buffer, const char \*format, [ argument] … );  
参数列表

buffer：char型指针，指向将要写入的字符串的缓冲区。  
format：格式化字符串。  
[argument]...：可选参数，可以是任何类型的数据。  
返回值：字符串长度（strlen）

2.转换字符

%% 印出百分比符号，不转换。  
%c 整数转成对应的 ASCII 字元。  
%d 整数转成十进位。  
%f 倍精确度数字转成浮点数。  
%o 整数转成八进位。  
%s 整数转成字符串。  
%x 整数转成小写十六进位。  
%X 整数转成大写十六进位。

描述

C 库函数 int sprintf(char \*str, const char \*format, ...) 发送格式化输出到 str 所指向的字符串。

声明

下面是 sprintf() 函数的声明。

int sprintf(char \*str, const char \*format, ...)

参数

str -- 这是指向一个字符数组的指针，该数组存储了 C 字符串。

format -- 这是字符串，包含了要被写入到字符串 str 的文本。它可以包含嵌入的 format 标签，format 标签可被随后的附加参数中指定的值替换，并按需求进行格式化。format 标签属性是 %[flags][width][.precision][length]specifier，具体讲解如下：

|  |  |
| --- | --- |
| **specifier（说明符）** | **输出** |
| c | 字符 |
| d 或 i | 有符号十进制整数 |
| e | 使用 e 字符的科学科学记数法（尾数和指数） |
| E | 使用 E 字符的科学科学记数法（尾数和指数） |
| f | 十进制浮点数 |
| g | 自动选择 %e 或 %f 中合适的表示法 |
| G | 自动选择 %E 或 %f 中合适的表示法 |
| o | 有符号八进制 |
| s | 字符的字符串 |
| u | 无符号十进制整数 |
| x | 无符号十六进制整数 |
| X | 无符号十六进制整数（大写字母） |
| p | 指针地址 |
| n | 无输出 |
| % | 字符 |

|  |  |
| --- | --- |
| **flags（标识）** | **描述** |
| - | 在给定的字段宽度内左对齐，默认是右对齐（参见 width 子说明符）。 |
| + | 强制在结果之前显示加号或减号（+ 或 -），即正数前面会显示 + 号。默认情况下，只有负数前面会显示一个 - 号。 |
| (space) | 如果没有写入任何符号，则在该值前面插入一个空格。 |
| # | 与 o、x 或 X 说明符一起使用时，非零值前面会分别显示 0、0x 或 0X。 与 e、E 和 f 一起使用时，会强制输出包含一个小数点，即使后边没有数字时也会显示小数点。默认情况下，如果后边没有数字时候，不会显示显示小数点。 与 g 或 G 一起使用时，结果与使用 e 或 E 时相同，但是尾部的零不会被移除。 |
| 0 | 在指定填充 padding 的数字左边放置零（0），而不是空格（参见 width 子说明符）。 |

|  |  |
| --- | --- |
| **width（宽度）** | **描述** |
| (number) | 要输出的字符的最小数目。如果输出的值短于该数，结果会用空格填充。如果输出的值长于该数，结果不会被截断。 |
| \* | 宽度在 format 字符串中未指定，但是会作为附加整数值参数放置于要被格式化的参数之前。 |

|  |  |
| --- | --- |
| **.precision（精度）** | **描述** |
| .number | 对于整数说明符（d、i、o、u、x、X）：precision 指定了要写入的数字的最小位数。如果写入的值短于该数，结果会用前导零来填充。如果写入的值长于该数，结果不会被截断。精度为 0 意味着不写入任何字符。 对于 e、E 和 f 说明符：要在小数点后输出的小数位数。 对于 g 和 G 说明符：要输出的最大有效位数。 对于 s: 要输出的最大字符数。默认情况下，所有字符都会被输出，直到遇到末尾的空字符。 对于 c 类型：没有任何影响。 当未指定任何精度时，默认为 1。如果指定时不带有一个显式值，则假定为 0。 |
| .\* | 精度在 format 字符串中未指定，但是会作为附加整数值参数放置于要被格式化的参数之前。 |

|  |  |
| --- | --- |
| **length（长度）** | **描述** |
| h | 参数被解释为短整型或无符号短整型（仅适用于整数说明符：i、d、o、u、x 和 X）。 |
| l | 参数被解释为长整型或无符号长整型，适用于整数说明符（i、d、o、u、x 和 X）及说明符 c（表示一个宽字符）和 s（表示宽字符字符串）。 |
| L | 参数被解释为长双精度型（仅适用于浮点数说明符：e、E、f、g 和 G）。 |

**附加参数** -- 根据不同的 format 字符串，函数可能需要一系列的附加参数，每个参数包含了一个要被插入的值，替换了 format 参数中指定的每个 % 标签。参数的个数应与 % 标签的个数相同。

返回值

如果成功，则返回写入的字符总数，不包括字符串追加在字符串末尾的空字符。如果失败，则返回一个负数。

**sprintf函数**

原型：int sprintf( char \*buffer, const char \*format, [ argument] … ); sprintf函数打印到字符串中

**usleep()函数**

usleep()函数是把调用该函数的线程挂起一段时间，单位是微秒（百万分之一秒）。

**unlink()函数**

unlink()函数功能即为删除文件，原型：int unlink(const char \*pathname);

### （4）输入输出函数

#### 2.1 单字符输入函数getchar()

1. getchar( )函数没有参数，有返回值，返回的就是输入的那个字符。
2. getchar( )函数一次只能接收一个字符。
3. getchar( )函数同样将空格和回车键等字符都会作为有效字符输入，所以注意和fflush(stdin)函数的结合使用。

#### 2.2 单字符输出函数putchar()

1. putchar（）函数有参数，无返回值。参数就是它要输出的那个字符，可以是字符变量或着字符常量
2. putchar（）函数一次向显示器输出一个字符。

#### 3.1 gets()/gets\_s()

用来接收用户输入的字符串，这是个有参数的函数，参数是一个字符数组类型，输入的字符串通过传址的方式传到这个字符数组中。

注意：在C++11中，采用 gets\_s，原型：char \*gets\_s(char \*buffer,size\_t sizeInCharacters);

如：  
char buffer[21]; //20 chars + ‘\0’  
gets\_s(buffer,20); //写成21后，当输入21个字符回车之后，自动添加的’\0’,会要求分配存储空间的  
printf(“Your input was %s\n”,buffer);

#### 3.2 puts()

用来向用户显示一个字符串，它需要一个字符串参数，显示该参数并自动换行。如：

puts(“hello, world.”);与printf(“hello,world./n”);的效果相同，如果是puts(“hello.world./n”)则要输出两个换行。也就是说，puts( )中的字符串可以包含转义字符。

### （5）文件操作函数

**读[字符串函数](https://www.baidu.com/s?wd=%E5%AD%97%E7%AC%A6%E4%B8%B2%E5%87%BD%E6%95%B0&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)fgets**

fgets函数的功能是从指定的文件中读一个字符串到字符数组中，函数调用的形式为： fgets(字符数组名，n，文件指针)；zhidao 其中的n是一个正整数。表示从文件中读出的字符串不超过 n-1个字符。在读入的最后一个字符后加上串结束标志'\0'。例如：fgets(str,n,fp);的意义是从fp所指的文专件中读出n-1个字符送入 字符数组str中。

**fputs函数**

fputs函数的功能是向指定的文件写入一个字符串，其调用形式为： fputs(字符串，文件指针) 其中字符串可以是[字符串常量](https://www.baidu.com/s?wd=%E5%AD%97%E7%AC%A6%E4%B8%B2%E5%B8%B8%E9%87%8F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)，也可以是字符数组名， 或指针变量.  
整块数据的读写函数。 可用来读写一组数据，如一个数组元素，一个结构变量的值等。

### （6）字符串操作函数

**1、strcmp函数**

strcmp函数是比较两个字符串的大小,返回比较的结果。一般形式是：   
i=strcmp(字符串，字符串);

其中，字符串１、字符串２均可为字符串常量或变量；i  是用于存放比较结果的整型变量。比较结果是这样规定的：   
①字符串1小于字符串2,strcmp函数返回一个负值;  
②字符串1等于字符串2,strcmp函数返回零;  
③字符串1大于字符串2,strcmp函数返回一个正值;

strcmp函数实际上是对字符的ASCII码进行比较，实现原理如下：首先比较两个串的第一个字符，若不相等，则停止比较并得出两个ASCII码大小比较的结果；如果相等就接着 比较第二个字符然后第三个字符等等。无论两个字符串是什么样，strcmp函数最多比较到其中一个字符串遇到结束符'/0'为止，就能得出结果。strcmp算法的可以有多种，不过我觉的可以把这么多算法分为两种，一种是利用减法运算判断结果，另一种是利用比较运算（==）得出结果。

注意:字符串是数组类型而非简单类型，不能用关系运算进行大小比较。

例：

if("ABC">"DEF")  /\*错误的字符串比较\*/  
if(strcmp("ABC","DEF")  /\*正确的字符串比较\*/

**2、strcpy函数**   
strcpy函数用于实现两个字符串的拷贝。一般形式是：   
strcpy（字符中1,字符串2)

其中,字符串1必须是字符串变量,而不能是字符串常量。strcpy函数把字符串2的内容完全复制到字符串1中,而不管字符串1中原先存放的是什么。复制后，字符串2保持不变。   
例:注意,由于字符串是数组类型,所以两个字符串复制不通过赋值运算进行。   
t=s;  /\*错误的字符串复制\*/  
strcpy(t,s);  /\*正确的字符串复制\*/

字符串处理strcpy strcat函数的用法：

1)strcat是用来连接两个字符串的，原型是char \*strcat(char \*dest,char \*src)，

作用是把src所指字符串添加到dest结尾处(覆盖dest结尾处的'\0')并添加'\0'

2) strcpy是用来把字符串拷贝到指定的地方的，

原型是char \*strcpy(char \*dest,const char \*src)，

作用是把从src地址开始且含有NULL结束符的字符串复制到以dest开始的地址空间

注意strcat是从dest的结尾处开始操作的，而strcpy是直接覆盖dest指向的内容。

**标准输出流**

它展开到一个 FILE\* （“指向 FILE 的指针”）类型的表达式（不一定是常量），

这个表达式指向一个与标准输出流（standard output stream）相关连的 FILE 对象

例子：

#include<stdio.h>

void func(FILE\*p)

{

char\*str="ddd";

fputs(str,p);

}

int main()

{

func(stdout);

return 0;

}

### （7）获取本地时间

**time() 函数、localtime() 函数、** **Demo**

**参考链接：**<https://www.cnblogs.com/zhaoosheLBJ/p/9322400.html>

<https://blog.csdn.net/llzhang_fly/article/details/82822908>

<https://www.cnblogs.com/wenqiang/p/5678451.html>

### （8）String家族(处理字符串的)

链接：

<https://blog.csdn.net/weixin_40519315/article/details/104125197?depth_1-utm_source=distribute.pc_relevant.none-task-blog-OPENSEARCH-8&utm_source=distribute.pc_relevant.none-task-blog-OPENSEARCH-8>

**strcpy,strcmp,strchr ,strstr,sprintf,strcat**

它们专门处理字符串

C语言是没有字符串类型的,它实际上就是一个字符数组

下面是它们的成员

登场吧,你认识它们几个?

strcpy,strcmp,strchr ,strstr,sprintf,strcat

**strcpy**

它是字符串拷贝函数,可以这么叫它(string copy)

它有个特点那就是 完全覆盖

下面举个栗子

**#include <stdio.h>**

**#include <string.h>**

**typedef char Str[20];**

**int main(int argc,char\*\* argv)**

**{**

**Str str1 ;**

**Str str2 ;**

**memset(str1,0,sizeof(str1));**

**memset(str2,0,sizeof(str2));**

**printf("str1<===str2\n");//str1将被str2覆盖掉所有**

**printf("str1=");**

**scanf("%s",str1);**

**printf("str2=");**

**scanf("%s",str2);**

**printf("str1=%s\n",strcpy(str1,str2));**

**return 0;**

**}**

输出:

str1<===str2

str1=hello

str2=sevenrainbow

str1=sevenrainbow

可见

str2的值全部覆盖到str1中

另外

还有一个它的另一个版本

**strncpy(string n copy)**

顾名思义,它可以指定多少进行覆盖

最后一个参数指的是 覆盖内容的大小(单位:字节)

strncpy(str1,str2,2);

这里,指定str2里开头的2个字节大小内容 覆盖掉  str1里开头的2个字节大小内容

输出结果如下:

str1<===str2

str1=12345

str2=ABCD

str1=AB345

**strcmp**

它是字符串比较函数 ,也可以叫 (string  compare)

很多时候我们都要比较两个字符串

这个时候用这个再好不过

我们来看下怎么用?

设这两个字符串为str1，str2，

若str1==str2，则返回零；

若str1<str2，则返回负数；

若str1>str2，则返回正数。

两个字符串自左向右逐个字符相比（按ASCII值大小相比较），直到出现不同的字符或遇'\0'为止。

下面举个栗子

**#include <stdio.h>**

**#include <string.h>**

**typedef char str\_20[20];**

**int main(int argc,char\*\* argv)**

**{**

**str\_20 str1,str2;**

**scanf("%s",str1);**

**scanf("%s",str2);**

**if(0==strcmp(str1,str2)){**

**printf("相同\n");**

**}else if(strcmp(str1,str2)){**

**printf("str1<str2\n");**

**}else{**

**printf("str1>str2\n");**

**}**

**}**

输出:

sevenRainbow

sevenRainbow

相同

sevenRainbow

sixRainbow

str1<str2

sixRainbow

sevenRainbow

str1<str2

**strchr**

在str里查找到指定chr并返回chr的地址

常用于关键字匹配、检索等

举个栗子:

**#include <stdio.h>**

**#include <string.h>**

**int main(int argc,char\*\* argv)**

**{**

**char \* p = NULL; //防止野指针（指针在没有初始化的时候它具有不确定性瞎指）**

**if(p = strchr("123@abc",'@'))**

**printf("found : %p %s \n",p,p+1);**

**else**

**printf("not found : %p \n",p);**

**return 0;**

**}**

解析：

if(p = strchr("123@abc",'@'))   小括号里赋值是什么情况？

看p最后的值是真是假即可，strchr的返回值是一个指针

放回的指针要么是一个地址（真），要么是NULL（假）。

**strstr**

在str里查找到指定str1并返回str1的地址

和strchr不同,它是检索字符串

常用于关键字匹配、检索等

范例：我们常用的播放器软件的工作原理：用检索查看目标文件的格式，然后再用相应的编码器进行解码播放。

**sprintf**

这个sprintf拥有一颗宽广的包容心

能够包容各种格式(整型、字符、浮点数等)

并将它们重新组织成新的字符串

在实际中,用处大大

**strcat**

将两个str相加

头接尾

### （9）switch用法

switch case用法详解

switch 是“开关”的意思，它也是一种“选择”语句，但它的用法非常简单。switch 是多分支选择语句。说得通俗点，多分支就是多个 if。

从功能上说，switch 语句和 if 语句完全可以相互取代。但从编程的角度，它们又各有各的特点，所以至今为止也不能说谁可以完全取代谁。

当嵌套的 if 比较少时（三个以内），用 if 编写程序会比较简洁。但是当选择的分支比较多时，嵌套的 if 语句层数就会很多，导致程序冗长，可读性下降。因此C语言提供 switch 语句来处理多分支选择。所以 if 和 switch 可以说是分工明确的。在很多大型的项目中，多分支选择的情况经常会遇到，所以 switch 语句用得还是比较多的。

switch的一般形式如下：

switch (表达式)

{

    case 常量表达式1：    语句1

    case 常量表达式2：    语句2

       ┇

    case 常量表达式n：    语句n

    default:        语句n+1

}

说明：

1) switch 后面括号内的“表达式”必须是整数类型。也就是说可以是 int 型变量、char 型变量，也可以直接是整数或字符常量，哪怕是负数都可以。但绝对不可以是实数，float 型变量、double 型变量、小数常量通通不行，全部都是语法错误。

2) switch 下的 case 和 default 必须用一对大括号{}括起来。

3) 当switch后面括号内“表达式”的值与某个case后面的“常量表达式”的值相等时，就执行此case后面的语句。执行完一个case后面的语句后，流程控制转移到下一个case继续执行。如果你只想执行这一个case语句，不想执行其他case，那么就需要在这个case语句后面加上break，跳出switch语句。

再重申一下：switch是“选择”语句，不是“循环”语句。很多新手看到break就以为是循环语句，因为break一般给我们的印象都是跳出“循环”，但break还有一个用法，就是跳出switch。

4) 若所有的 case 中的常量表达式的值都没有与 switch 后面括号内“表达式”的值相等的，就执行 default 后面的语句，default 是“默认”的意思。如果 default 是最后一条语句的话，那么其后就可以不加 break，因为既然已经是最后一句了，则执行完后自然就退出 switch 了。

5) 每个 case 后面“常量表达式”的值必须互不相同，否则就会出现互相矛盾的现象，而且这样写造成语法错误。

6) “case常量表达式”只是起语句标号的作用，并不是在该处进行判断。在执行 switch 语句时，根据 switch 后面表达式的值找到匹配的入口标号，就从此标号开始执行下去，不再进行判断。

7) 各个 case 和 default 的出现次序不影响执行结果。但从阅读的角度最好是按字母或数字的顺序写。

8) 当然你也可以不要 default 语句，就跟 if…else 最后不要 else 语句一样。但最好是加上，后面可以什么都不写。这样可以避免别人误以为你忘了进行 default 处理，而且可以提醒别人 switch 到此结束了。

但是需要注意的是，default 后面可以什么都不写，但是后面的冒号和分号千万不能省略，省略了就是语法错误。很多新手在这个地方很容易出错，要么忘了分号，要么忘了冒号，所以要注意！

### （10）break

break为打断当前循环程序，跳到更高级中。

该例子为while循环

如果循环中有if函数，则直接跳出if语句重新回到循环中

如果if语言嵌套，则返回上一级语句

switch中，如果没有碰到break语句，则继续运行到遇到break或程序结束

usleep()函数是把调用该函数的线程挂起一段时间，单位是微秒（百万分之一秒）。

unlink()函数功能即为删除文件，原型：int unlink(const char \*pathname);

### 参考链接：

<https://www.runoob.com/cprogramming/c-standard-library-string-h.html>