strtok()—字符串分割函数

相关函数：index, memchr, rindex, strpbrk, strsep, strspn, strstr

头文件：#include <string.h>

定义函数：char * strtok(char *s, const char *delim);

函数说明：**strtok()**用来将字符串分割成一个个片段. 参数 **s** 指向欲分割的字符串, 参数 **delim** 则为分割字符串,当 **strtok()**在参数 **s** 的字符串中发现到参数 **delim** 的分割字符时则会将该字符改为**\0** 字符. 在第一次调用时,**strtok()**必需给予参数 **s** 字符串, 往后的调用则将参数 **s** 设置成 **NULL**. 每次调用成功则返回下一个分割后的字符串指针.

返回值：返回下一个分割后的字符串指针, 如果已无从分割则返回 NULL.

范例

```
#include <string.h>
main()
{
    char s[] = "ab-cd : ef;gh :i-jkl;mnop;qrs-tu: vwx-y;z";
    char *delim = "-: ";
    char *p;
    printf("%s ", strtok(s, delim));
    while((p = strtok(NULL, delim)))
        printf("%s ", p);
        printf("\n");
}
```

执行结果：

```
ab cd ef;gh i jkl;mnop;qrs tu vwx y;z //－与:字符已经被\0 字符取代
```

strstr()—字符串查找函数

相关函数：index, memchr, rindex, strchr, strpbrk, strsep, strspn, strtok

头文件：#include <string.h>

定义函数：char *strstr(const char *haystack, const char * needle);

函数说明：**strstr()**会从字符串 **haystack** 中搜寻字符串 **needle**, 并将第一次出现的地址返回.

返回值：返回指定字符串第一次出现的地址, 否则返回 0.

范例

```
#include <string.h>
main()
{
    char * s = "012345678901234567890123456789";
    char *p;
    p = strstr(s, "901");
    printf("%s\n", p);
}
```

执行结果：

9.01E+21

strspn()—字符查找函数

相关函数：strcspn, strchr, strpbrk, strsep, strstr

头文件：#include <string.h>

定义函数：size_t strspn(const char *s, const char * accept);

函数说明：**strspn()**从参数 **s** 字符串的开头计算连续的字符，而这些字符都完全是 **accept** 所指字符串中的字符.简单的说，若 **strspn()**返回的数值为 **n**，则代表字符串 **s** 开头连续有 **n** 个字符都是属于字符串 **accept** 内的字符.

返回值：返回字符串 s 开头连续包含字符串 accept 内的字符数目.

范例

```
#include <string.h>
main()
{
char *str = "Linux was first developed for 386/486-based PCs. ";
char *t1 = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
printf("%d\n", strspn(str, t1));
}
```

执行结果：

5 //计算大小写字母. 不包含" ", 所以返回 Linux 的长度.

strrchr()—定位字符串中最后出现的指定字符

相关函数：index, memchr, rindex, strpbrk, strsep, strspn, strstr, strtok

头文件：#include <string.h>

定义函数：char * strrchr(const char *s, int c);

函数说明：**strrchr()**用来找出参数 **s** 字符串中最后一个出现的参数 **c** 地址，然后将该字符出现的地址返回.

返回值：如果找到指定的字符则返回该字符所在地址，否则返回 0.

范例

```
#include <string.h>
main()
{
    char *s = "01234567890123456789012345678901234567890";
    char *p;
    p = strrchr(s, '5');
    printf("%s\n", p);
}
```

执行结果：

567890

strpbrk()—定位字符串中第一个出现的指定字符

相关函数：index, memchr, rindex, strpbrk, strsep, strspn, strstr, strtok

头文件：#include <include.h>

定义函数：char *strpbrk(const char *s, const char *accept);

函数说明：**strpbrk()**用来找出参数 **s** 字符串中最先出现存在参数 **accept** 字符串中的任意字符.

返回值：如果找到指定的字符则返回该字符所在地址，否则返回 0.

范例

```
#include <string.h>
main()
{
    char *s = "01234567890123456789012345678901234567890";
    char *p; p = strpbrk(s, "a1 839"); //1 会最先在 s 字符串中找到
    printf("%s\n", p);
```

```
    p = strprk(s, "4398"); //3 会最先在 s 字符串中找到
    printf("%s\n", p);
}
```

执行结果：

```
1.23E+29
```

strncat()—字符串连接函数

相关函数：bcopy, memccpy, memecpy, strcpy, strncpy

头文件：#inclue <string.h>

定义函数：char * strncat(char *dest, const char *src, size_t n);

函数说明：**strncat()**会将参数 **src** 字符串拷贝 **n** 个字符到参数 **dest** 所指的字符串尾. 第一个参数 **dest** 要有足够的空间来容纳要拷贝的字符串.

返回值：返回参数 dest 的字符串起始地址.

范例

```
#include <string.h>
main()
{
    char a[30] = "string(1)";
    char b[] = "string(2)";
    printf("before strncat() :%s\n", a);
    printf("after strncat() :%s\n", strncat(a, b, 6));
}
```

执行结果：

```
before strncat() : string(1)
after strncat() : string(1) string
```

strncpy()—复制字符串

相关函数：bcopy, memccpy, memcpy, memmove

头文件：#include <string.h>

定义函数：char * strncpy(char *dest, const char *src, size_t n);

函数说明：**strncpy()**会将参数 **src** 字符串拷贝前 **n** 个字符至参数 **dest** 所指的地址.

返回值：返回参数 dest 的字符串起始地址.

范例

```
#inclue <string.h>
main()
{
    char a[30] = "string(1)";
    char b[] = "string(2)";
    printf("before strncpy() : %s\n", a);
    printf("after strncpy() : %s\n", strncpy(a, b, 6));
}
```

执行结果：

```
before strncpy() : string(1)
after strncpy() : string(1)
```

strncasecmp()—字符串比较函数(忽略大小写)

相关函数：bcmp, memcmp, strcmp, strcoll, strncmp

头文件：#include <string.h>

定义函数：int strncasecmp(const char *s1, const char *s2, size_t n);

函数说明：**strncasecmp()**用来比较参数 **s1** 和 **s2** 字符串前 **n** 个字符，比较时会自动忽略大小写的差异.

返回值：若参数 s1 和 s2 字符串相同则返回 0. s1 若大于 s2 则返回大于 0 的值, s1 若小于 s2 则返回小于 0 的值.

范例

```
#include <string.h>
main()
{
    char *a = "aBcDeF";
    char *b = "AbCdEf";
    if(!strncasecmp(a, b))
    printf("%s =%s\n", a, b);
}
```

执行结果：

```
aBcDef=AbCdEf
```

strlen()—字符串长度计算函数

相关函数：无

头文件：#include <string.h>

定义函数：size_t strlen (const char *s);

函数说明：**strlen()**用来计算指定的字符串 **s** 的长度，不包括结束字符**"\0".**

返回值：返回字符串 s 的字符数.

范例：

```
/*取得字符串 str 的长度 */
#include <string.h>
main()
{
    char *str = "12345678";
    printf("str length = %d\n", strlen(str));
}
```

执行结果：

```
str length = 8
```

strdup()—复制字符串

相关函数：calloc, malloc, realloc, free

头文件：#include <string.h>

定义函数：char * strdup(const char *s);

函数说明：**strdup()**会先用 **maolloc()**配置与参数 **s** 字符串相同的空间大小，然后将参数 **s** 字符串的内容复制到该内存地址，然后把该地址返回. 该地址最后可以利用 **free()**来释放.

返回值：返回一字符串指针，该指针指向复制后的新字符串地址. 若返回 NULL 表示内存不足.

范例

```
#include <string.h>
main()
```

```
{
    char a[] = "strdup";
    char *b;
    b = strdup(a);
    printf("b[]=\"%s\"\n", b);
}
```

执行结果：

```
b[]="strdup"
```

strcspn()—查找字符串

相关函数：strspn

头文件：#inclued<string.h>

定义函数：size_t strcspn(const char *s, const char * reject);

函数说明：**strcspn()**从参数 **s** 字符串的开头计算连续的字符，而这些字符都完全不在参数 **reject** 所指的字符串中. 简单地说，若 **strcspn()**返回的数值为 **n**，则代表字符串 **s** 开头连续有 **n** 个字符都不含字符串 **reject** 内的字符.

返回值：返回字符串 s 开头连续不含字符串 reject 内的字符数目.

范例

```
#include <string.h>
main()
{
    char *str = "Linux was first developed for 386/486-based pcs. ";
    printf("%d\n", strcspn(str, " "));
    printf("%d\n", strcspn(str, "/-"));
    printf("%d\n", strcspn(str, "1234567890"));
}
```

执行结果：

```
5 //只计算到" "的出现，所以返回"Linux"的长度
33 //计算到出现"/"或"－"，所以返回到"6"的长度
30 // 计算到出现数字字符为止，所以返回"3"出现前的长度
```

strcpy()—复制字符串

相关函数：bcopy, memcpy, memccpy, memmove

头文件：#include <string.h>

定义函数：char *strcpy(char *dest, const char *src);

函数说明：strcpy()会将参数 src 字符串拷贝至参数 dest 所指的地址.

返回值：返回参数 dest 的字符串起始地址.

附加说明：如果参数 **dest** 所指的内存空间不够大，可能会造成缓冲溢出**(buffer Overflow)**的错误情况，在编写程序时请特别留意，或者用 **strncpy()**来取代.

范例

```
#include <string.h>
main()
{
    char a[30] = "string(1)";
    char b[] = "string(2)";
    printf("before strcpy() :%s\n", a);
    printf("after strcpy() :%s\n", strcpy(a, b));
}
```

执行结果：

```
before strcpy() :string(1)
after strcpy() :string(2)
```

strcoll()—字符串比较函数(按字符排列次序)

相关函数：strcmp, bcmp, memcmp, strcasecmp, strncasecmp

头文件：#include <string.h>

定义函数：int strcoll(const char *s1, const char *s2);

函数说明：**strcoll()**会依环境变量 **LC_COLLATE** 所指定的文字排列次序来比较 **s1** 和 **s2** 字符串.

返回值：若参数 s1 和 s2 字符串相同则返回 0. s1 若大于 s2 则返回大于 0 的值. s1 若小于 s2 则返回小于 0 的值.

附加说明：若 **LC_COLLATE** 为**"POSIX"**或**"C"**，则 **strcoll()**与 **strcmp()**作用完全相同.

范例 参考 strcmp().

strcmp()—字符串比较函数(比较字符串)

相关函数：bcmp, memcmp, strcasecmp, strncasecmp, strcoll

头文件：#include <string.h>

定义函数：int strcmp(const char *s1, const char *s2);

函数说明：**strcmp()**用来比较参数 **s1** 和 **s2** 字符串. 字符串大小的比较是以 **ASCII** 码表上的顺序来决定, 此顺序亦为字符的值. **strcmp()**首先将 **s1** 第一个字符值减去 **s2** 第一个字符值, 若差值为 **0** 则再继续比较下个字符, 若差值不为 **0** 则将差值返回. 例如字符串 **"Ac"**和**"ba"**比较则会返回字符**"A"(65)**和**'b'(98)**的差值**(−33).**

返回值：若参数 s1 和 s2 字符串相同则返回 0. s1 若大于 s2 则返回大于 0 的值. s1 若小于 s2 则返回小于 0 的值.

范例

```
#include <string.h>
main()
{
    char *a = "aBcDeF";
    char *b = "AbCdEf";
    char *c = "aacdef";
    char *d = "aBcDeF";
    printf("strcmp(a, b) : %d\n", strcmp(a, b));
    printf("strcmp(a, c) : %d\n", strcmp(a, c));
    printf("strcmp(a, d) : %d\n", strcmp(a, d));
}
```

执行结果：

```
strcmp(a, b) : 32
strcmp(a, c) :-31
strcmp(a, d) : 0
```

strchr()—字符串查找函数(返回首次出现字符的位置)

相关函数：index, memchr, rinex, strbrk, strsep, strspn, strstr, strtok

头文件：#include <string.h>

定义函数：char * strchr (const char *s, int c);

函数说明：**strchr()**用来找出参数 **s** 字符串中第一个出现的参数 **c** 地址, 然后将该字符出

现的地址返回.

返回值：如果找到指定的字符则返回该字符所在地址，否则返回 0.

范例

```
#include <string.h>
main()
{
    char *s = "012345678901234567890123456789O";
    char *p;
    p = strchr(s, '5');
    printf("%s\n", p);
}
```

执行结果：

5.68E+25

strcat()—连接字符串

相关函数：bcopy, memccpy, memcpy, strcpy, strncpy

头文件：#include <string.h>

定义函数：char *strcat(char *dest, const char *src);

函数说明：**strcat()**会将参数 **src** 字符串拷贝到参数 **dest** 所指的字符串尾. 第一个参数 **dest** 要有足够的空间来容纳要拷贝的字符串.

返回值：返回参数 dest 的字符串起始地址

范例

```
#include <string.h>
main()
{
    char a[30] = "string(1)";
    char b[] = "string(2)";
    printf("before strcat() : %s\n", a);
    printf("after strcat() : %s\n", strcat(a, b));
}
```

执行结果：

before strcat() : string(1)
after strcat() : string(1)string(2)

strcasecmp()—字符串比较函数(忽略大小写比较字符串)

相关函数：bcmp, memcmp, strcmp, strcoll, strncmp

头文件：#include <string.h>

定义函数：int strcasecmp (const char *s1, const char *s2);

函数说明：**strcasecmp()**用来比较参数 **s1** 和 **s2** 字符串，比较时会自动忽略大小写的差异.

返回值：若参数 s1 和 s2 字符串相同则返回 0. s1 长度大于 s2 长度则返回大于 0 的值, s1 长度若小于 s2 长度则返回小于 0 的值.

范例

```
#include <string.h>
main()
{
    char *a = "aBcDeF";
    char *b = "AbCdEf";
    if(!strcasecmp(a, b))
    printf("%s=%s\n", a, b);
}
```

执行结果：

aBcDeF=AbCdEf

rindex()—字符串查找函数(返回最后一次出现的位置)

相关函数：index, memchr, strchr, strrchr

头文件：#include <string.h>

定义函数：char * rindex(const char *s, int c);

函数说明：**rindex()**用来找出参数 **s** 字符串中最后一个出现的参数 **c** 地址，然后将该字符出现的地址返回. 字符串结束字符**(NULL)**也视为字符串一部分.

返回值：如果找到指定的字符则返回该字符所在的地址，否则返回 0.

范例

```
#include <string.h>
main()
{
    char *s = "01234567890123456789012345678890";
    char *p;
    p = rindex(s, '5');
    printf("%s\n", p);
}
```

执行结果：

567890

index()—字符串查找函数(返回首次出现的位置)

相关函数：rindex, srechr, strrchr

头文件：#include <string.h>

定义函数：char * index(const char *s, int c);

函数说明：**index()**用来找出参数 **s** 字符串中第一个出现的参数 **c** 地址，然后将该字符出现的地址返回。字符串结束字符**(NULL)**也视为字符串一部分。

返回值：如果找到指定的字符则返回该字符所在地址，否则返回 0.

范例

```
#include <string.h>
main()
{
    char *s = "01234567890123456789012345678890";
    char *p;
    p = index(s, '5');
    printf("%s\n", p);
}
```

执行结果：

5.68E+25

toupper()—字符串转换函数(小写转大写)

相关函数：isalpha, tolower

头文件：#include <ctype.h>

定义函数：int toupper(int c);

函数说明：若参数 **c** 为小写字母则将该对应的大写字母返回.

返回值：返回转换后的大写字母, 若不须转换则将参数 c 值返回.

范例 /* 将 s 字符串内的小写字母转换成大写字母 */

```
#include <ctype.h>
main()
{
    char s[] = "aBcDeFgH12345;!#$";
    int i;
    printf("before toupper() : %s\n", s);
    for(i = 0; I < sizeof(s); i++)
        s[i] = toupper(s[i]);
        printf("after toupper() : %s\n", s);
}
```

执行结果：

```
before toupper() : aBcDeFgH12345;!#$
after toupper() : ABCDEFGH12345;!#$
```

tolower()—字符串转换函数(大写转小写)

作

相关函数：isalpha, toupper

头文件：#include <stdlib.h>

定义函数：int tolower(int c);

函数说明：若参数 **c** 为大写字母则将该对应的小写字母返回.

返回值：返回转换后的小写字母, 若不须转换则将参数 c 值返回.

范例

```
/* 将 s 字符串内的大写字母转换成小写字母 */
#include <ctype.h>
main()
{
```

```
    char s[] = "aBcDeFgH12345;!#$";
    int i;
    printf("before tolower() : %s\n", s);
    for(i = 0; I < sizeof(s); i++)
        s[i] = tolower(s[i]);
        printf("after tolower() : %s\n", s);
}
```

执行结果：

before tolower() : aBcDeFgH12345;!#$

after tolower() : abcdefgh12345;!#$

toascii()—将整数转换成合法的 ASCII 码字符

相关函数：isascii, toupper, tolower

头文件：#include <ctype.h>

定义函数：int toascii(int c);

函数说明：**toascii()**会将参数 **c** 转换成 **7** 位的 **unsigned char** 值，第八位则会被清除，此字符即会被转成 **ASCII** 码字符.

返回值：将转换成功的 ASCII 码字符值返回.

范例

```
/* 将 int 型 a 转换成 ASSII 码字符 */
#include <stdlib.h>
main()
{
    int a = 217;
    char b;
    printf("before toascii() : a value =%d(%c)\n", a, a);
    b = toascii(a);
    printf("after toascii(): a value =%d(%c)\n", b, b);
}
```

执行结果：

before toascii() : a value =217()

after toascii() : a value =89(Y)

strtoul()—将字符串转换成无符号长整型数

相关函数：atof, atoi, atol, strtod, strtol

头文件：#include <stdlib.h>

定义函数：unsigned long int strtoul(const char *nptr, char **endptr, int base);

函数说明：
strtoul()会将参数**nptr** 字符串根据参数**base** 来转换成无符号的长整型数。参数**base** 范围从 **2** 至 **36,** 或 **0**;参数 **base** 代表采用的进制方式，如 **base** 值为 **10** 则采用 **10** 进制，若 **base** 值为 **16** 则采用 **16** 进制数等。当 **base** 值为 **0** 时则是采用 **10** 进制做转换，但遇到如**'0x'**前置字符则会使用 **16** 进制做转换。一开始 **strtoul()**会扫描参数 **nptr** 字符串，跳过前面的空格字符串，直到遇上数字或正负符号才开始做转换，再遇到非数字或字符串结束时**('\0')**结束转换，并将结果返回。若参数 **endptr** 不为 **NULL,** 则会将遇到不合条件而终止的 **nptr** 中的字符指针由 **endptr** 返回。

返回值：返回转换后的长整型数，否则返回 ERANGE 并将错误代码存入 errno 中.

附加说明：ERANGE 指定的转换字符串超出合法范围.

范例 参考 strtol()

strtol()—将字符串转换成长整型数

相关函数：atof, atoi, atol, strtod, strtoul

头文件：#include <stdlib.h>

定义函数：long int strtol(const char *nptr, char **endptr, int base);

函数说明：
strtol()会将参数**nptr** 字符串根据参数**base** 来转换成长整型数。参数**base** 范围从**2** 至 **36,** 或 **0.**参数 **base** 代表采用的进制方式，如 **base** 值为 **10** 则采用 **10** 进制，若 **base** 值为 **16** 则采用 **16** 进制等。当 **base** 值为 **0** 时则是采用 **10** 进制做转换，但遇到如**'0x'** 前置字符则会使用 **16** 进制做转换。一开始 **strtol()**会扫描参数 **nptr** 字符串，跳过前面的空格字符，直到遇上数字或正负符号才开始做转换，再遇到非数字或字符串结束时**('\0')**结束转换，并将结果返回。若参数 **endptr** 不为 **NULL,** 则会将遇到不合条件而终止的 **nptr** 中的字符指针由 **endptr** 返回。

返回值：返回转换后的长整型数，否则返回 ERANGE 并将错误代码存入 errno 中.

附加说明：**ERANGE** 指定的转换字符串超出合法范围。

范例

```
/* 将字符串 a, b, c 分别采用 10, 2, 16 进制转换成数字 */
#include <stdlib.h>
main()
{
    char a[] = "1000000000";
    char b[] = "1000000000";
    char c[] = "ffff";
    printf("a=%d\n", strtol(a, NULL, 10));
    printf("b=%d\n", strtol(b, NULL, 2));
    printf("c=%d\n", strtol(c, NULL, 16));
}
```

执行结果：

```
a=1000000000
b=512
c=65535
```

strtod()—将字符串转换成浮点数

相关函数：atoi, atol, strtod, strtol, strtoul

头文件：#include <stdlib.h>

定义函数：double strtod(const char *nptr, char **endptr);

函数说明：

strtod()会扫描参数 **nptr** 字符串，跳过前面的空格字符，直到遇上数字或正负符号才开始做转换，到出现非数字或字符串结束时**('\0')**才结束转换，并将结果返回。若 **endptr** 不为 **NULL**，则会将遇到不合条件而终止的 **nptr** 中的字符指针由 **endptr** 传回。参数 **nptr** 字符串可包含正负号、小数点或 **E(e)**来表示指数部分。如 **123. 456** 或 **123e-2.**

返回值：返回转换后的浮点型数.

附加说明：参考 **atof().**

范例

```
/*将字符串 a, b, c 分别采用 10, 2, 16 进制转换成数字 */
#include <stdlib.h>
main()
{
```

```
    char a[] = "1000000000";
    char b[] = "1000000000";
    char c[] = "ffff";
    printf("a=%d\n", strtod(a, NULL, 10));
    printf("b=%d\n", strtod(b, NULL, 2));
    printf("c=%d\n", strtod(c, NULL, 16));
}
```

执行结果：

```
a=1000000000
b=512 c=65535
```

gcvt()—将浮点型数转换为字符串(四舍五入)

相关函数：ecvt, fcvt, sprintf

头文件：#include <stdlib.h>

定义函数：char *gcvt(double number, size_t ndigits, char *buf);

函数说明：

gcvt()用来将参数 **number** 转换成 **ASCII** 码字符串，参数 **ndigits** 表示显示的位数。**gcvt()**与 **ecvt()**和 **fcvt()**不同的地方在于，**gcvt()**所转换后的字符串包含小数点或正负符号。若转换成功，转换后的字符串会放在参数 **buf** 指针所指的空间。

返回值：返回一字符串指针，此地址即为 buf 指针。

范例

```
#include <stdlib.h>
main()
{
    double a = 123.45;
    double b = -1234.56;
    char *ptr;
    int decpt, sign;
    gcvt(a, 5, ptr);
    printf("a value=%s\n", ptr);
    ptr = gcvt(b, 6, ptr);
    printf("b value=%s\n", ptr);
}
```

执行结果：

a value=123.45
b value=-1234.56

atol()—将字符串转换成长整型数

相关函数：atof, atoi, strtod, strtol, strtoul

头文件：#include <stdlib.h>

定义函数：long atol(const char *nptr);

函数说明：**atol()**会扫描参数 **nptr** 字符串，跳过前面的空格字符，直到遇上数字或正负符号才开始做转换，而再遇到非数字或字符串结束时**('\0')**才结束转换，并将结果返回**.**

返回值：返回转换后的长整型数.

附加说明：**atol()**与使用 **strtol(nptr, (char**)NULL, 10);** 结果相同**.**

范例

```
/*将字符串 a 与字符串 b 转换成数字后相加 */
#include <stdlib.h>
main()
{
    char a[] = "1000000000";
    char b[] = " 234567890";
    long c;
    c = atol(a) + atol(b);
    printf("c=%d\n", c);
}
```

执行结果：

c=1234567890

atoi()—将字符串转换成整型数

相关函数：atof, atol, atrtod, strtol, strtoul

头文件：#include <stdlib.h>

定义函数：int atoi(const char *nptr);

函数说明：**atoi()**会扫描参数 **nptr** 字符串，跳过前面的空格字符，直到遇上数字或正负符号才开始做转换，而再遇到非数字或字符串结束时**('\0')**才结束转换，并将结果返回**.**返回

值返回转换后的整型数.

附加说明：atoi()与使用 strtol(nptr, (char**)NULL, 10); 结果相同.

范例

```
/* 将字符串 a 与字符串 b 转换成数字后相加 */
#include <stdlib.h>
main()
{
    char a[] = "-100";
    char b[] = "456";
    int c;
    c = atoi(a) + atoi(b);
    printf("c=%d\n", c);
}
```

执行结果：

```
c=356
```

atof()—将字符串转换成浮点型数

相关函数：atoi, atol, strtod, strtol, strtoul

头文件：#include <stdlib.h>

定义函数：double atof(const char *nptr);

函数说明：**atof()**会扫描参数 **nptr** 字符串，跳过前面的空格字符，直到遇上数字或正负符号才开始做转换，而再遇到非数字或字符串结束时**('\0')**才结束转换，并将结果返回. 参数 **nptr** 字符串可包含正负号、小数点或 **E(e)**来表示指数部分，如 **123. 456** 或 **123e-2.**

返回值：返回转换后的浮点型数.

附加说明：**atof()**与使用 **strtod(nptr, (char**)NULL)**结果相同.

范例：

```
/* 将字符串 a 与字符串 b 转换成数字后相加 */
#include <stdlib.h>
main()
{
    char *a = "-100.23";
    char *b = "200e-2";
```

```
    float c;
    c = atof(a) + atof(b);
    printf("c=%.2f\n", c);
}
```

执行结果：

```
c=-98.23
```