# 1、strstr

函数原型：char \*strstr(const char \*haystack, const char \*needle);

头文件：#include <string.h>

函数功能：在haystack中查找needle第一次出现的位置，若找到，返回当前char \*指针，否则返回NULL。

注：严格区分大小写。

不区分大小写的是下面这个。

strctr (const char \*haystack, int c);查找haystack中第一次出现字符无符号字符c的位置。如找到返回第一次出现的

字符的指针，如果字符串中不包含，则返回NULL/

# 2、strcasestr

和上面一致，不区分大小写的。

# 3、fread

函数原型：  fread(   void   \*buffer,   size\_t   size,   size\_t   count,   FILE   \*stream   )

头文件：#include<stdio.h>

函数功能：从stream中读取count次size大小的数据存放到buffer中。

# 4、unlink

函数原型：int unlink(const char \*pathname)

头文件：#include<unistd.h>

函数功能：删除参数所给文件。成功返回0，失败返回-1；

注：执行Unlink函数并不一定会删除文件，他会先检查文件系统中此文件的连接数是否为1，如果不是1，说明此文件还有其他链接对象，此时unlink函数之后对文件的连接数减一，并不会真正的删除，直到所有该文件打开的进程都结束的时候才会被真正删除。若连接数为1，并且在此时没有任何进程打开该文件，此文件才会被真正的删除掉。

# 5、static

静态全局变量只能在当前定义的源文件中使用，不可被外部引用。如果在头文件中定义静态全局，则所有包含头文件的都可以使用这个变量，但是只是看起来使用的是同一个变量，但是使用的是不同的变量，地址空间不同。只是定义了很多次而已，不是真正的全局变量，造成空间的浪费。应该及其避免使用在头文件中定义静态全局变量的形式。

正确使用：在源文件中定义变量（非静态），在其他需要引用的地方使用extern。

函数前面使用static，表明函数是被当前文件作用，不可被其他文件引用，失去全局可见性。

注：使用extern的时候，不可以跟初值，否则，还是变量定义，会出现重定义的错误。

# 6、strspn

返回字符串中第一个不在指定字符串中出现的字符下标。

strspn(const char \*s,const char \*a); 如果strcpn返回值为n，则代表字符串s开头有个字符都是属于字符串a中的内容。

附：读取文件中某一行的内容。

int main()

{

FILE \*fp = NULL;

char cBuffer[128] = {0};

char \*ptr = NULL;

fp = fopen("readme.ini", "r");

if(fp)

{

while (fgets(cBuffer, sizeof(cBuffer)/sizeof(char), fp))

{

if (strchr(cBuffer, '\n'))

{

\*(strchr(cBuffer, '\n')) = '\0';

}

if('\0' == cBuffer[0])

{

continue;

}

ptr = cBuffer + strspn(cBuffer, " \t");//消去字符前空格

if ('\0' == \*ptr)

{

ptr = NULL;

continue;

}

if(!strncmp(ptr,"abc",sizeof("abc")))

{

printf("yanzhentao \n");

}

}

}

}

# 7、remove（）

int remove(char \* filename);

参数filename为要删除的文件名，可以为目录，也可以为文件。

当为目录的时候，调用rmdir(),

当为文件的时候会调用unlink();

成功返回0，失败返回-1.错误原因存在与errno

# 8、gettimeofday time

需要包含头文件#include<time.h>

1、gettimeofday（struct timeval \*begintime,struct tiimezone \*tz）

返回当前距离1970年1月1日的秒数和毫秒数。tz为时区，一般为0.

timeval

{

long tv\_sec; //秒

long tv\_usec;//微秒

}

一般转换为毫秒。tv\_sec\*1000+tv\_usec/1000

2、time函数，将整数变为人类可读的时间。

time\_t 是一个长整数，表示从1970-01-01 00:00:00到目前的秒数。

tm 结构，本身是个结构体，里面包含了各种时间段。  
struct tm {

int tm\_sec; /\* seconds after the minute - [0,59] \*/

int tm\_min; /\* minutes after the hour - [0,59] \*/

int tm\_hour; /\* hours since midnight - [0,23] \*/

int tm\_mday; /\* day of the month - [1,31] \*/

int tm\_mon; /\* months since January - [0,11] \*/

int tm\_year; /\* years since 1900 \*/

int tm\_wday; /\* days since Sunday - [0,6] \*/

int tm\_yday; /\* days since January 1 - [0,365] \*/

int tm\_isdst; /\* daylight savings time flag \*/

};

一般

time\_t t ;//秒时间

tm \*local;

char buf[128] = {0};

t = time(NULL);//或者time(&t);//获取秒数

local = localtime(&t);//转换为本地时间

strftime(buf,64,”%Y-%m-%d %H:%M:%S”,local);//将时间格式化输出。

printf(“%s \n”,buf);

# 9、va\_list

va\_list在C语言中解决参数个数不确定问题的一组宏。

如：AveInt(int v,…)；v表示参数的个数。

（1）、先在函数中定义一个va\_list的变量，变量指向参数的指针；

（2）、va\_start宏初始化变量刚定义的va\_list变量。

（3）、va\_arg返回可变的参数，第二个参数是要返回的参数类型。

（4）、va\_end结束可变参数的获取。

其中va\_arg是返回可变的参数，一般是把传入的参数，格式化之后传给目的变量。这时候使用\_vsnprintf。

\_vsnprintf 把可变参数格式化输出到一个字符数组。

linux环境是vsnprintf

vc6环境下是\_vsnprintf

**int mon\_log(char\* format, ...)**

**{**

**char str\_tmp[50];**

**int i=0;**

**va\_list vArgList; /定义一个va\_list型的变量,这个变量是指向参数的指针.**

**va\_start (vArgList, format); //用va\_start宏初始化变量,这个宏的第二个参数是第一个可变参数的前一个参数,是一个固定的参数**

**i=vsnprintf(str\_tmp, 50, format, vArgList);**

**va\_end(vArgList); //用va\_end宏结束可变参数的获取**

**}**