来说一说fgets(..)函数。

原型 char * fgets(char * s, int n,FILE *stream);

参数:

s: 字符型指针，指向存储读入数据的缓冲区的地址。

n: 从流中读入n-1个字符

stream： 指向读取的流。

返回值:

1. 当n<=0 时返回NULL，即空指针。
2. 当n=1 时，返回空串"".
3. 如果读入成功，则返回缓冲区的地址。
4. 如果读入错误或遇到文件结尾(EOF)，则返回NULL.

看看这个函数的官方说明:

```
/***
*char *fgets(string, count, stream) - input string from a stream
*
*Purpose:
* get a string, up to count-1 chars or '\n', whichever comes first,
* append '\0' and put the whole thing into string. the '\n' IS included
* in the string. if count<=1 no input is requested. if EOF is found
* immediately, return NULL. if EOF found after chars read, let EOF
* finish the string as '\n' would.
*
```

```
*Entry:
* char *string - pointer to place to store string
* int count - max characters to place at string (include
\0)
* FILE *stream - stream to read from
*
*Exit:
* returns string with text read from file in it.
* if count <= 0 return NULL
* if count == 1 put null string in string
* returns NULL if error or end-of-file found
immediately
*
*Exceptions:
*

*******************************************************************************/
```

标准库中fgets(...)的实现：

```
/***************************************************
 char *fgets(char *s, int n,  FILE *stream)
 {
     register int c;
     register char *cs;
     cs=s;
     while(--n>0 &&(c = getc(stream))!=EOF)
         if ((*cs++=  c) =='\n')
           break;
```

```
        *cs ='\0';
        return (c == EOF && cs == s) ?NULL :s ;
      }
/*********************************************************
```

在用fgets(..)读入数据时，先定义一个字符数组或字符指针，如果定义了字符指针，那么一定要初始化。

example:

char s[100]; //可以。

char *s; //不可以，因为只是声明了一个指针。但并没有为它分配内存缓冲区。

所以，如果要用指针，则 char *s=(char *)malloc(100*sizeof(char)); 为其分配内存空间,c++中用char *s=new char [100]; 如果为分配内存空间，编译时不会检查出问题，但运行时会出现未知错误。。

fgets(...)读入文本行时的两种情况。

1。 如果n大于一行的字符串长度，那么当读到字符串末尾的换行符时，fgets(..)会返回。并且在s的最后插入字符串结束标志'\0'。 而s缓冲区剩余的位置不会再填充。

example：

123abc

fgets(s,10,fp);

此时，读入七个字符，123abc\n,实际上还有最后的'\0',所以，strlen(s)=7; 如果要去除末尾的\n，s[strlen(s)-1]='\0';便可。

2. 如果n小于等于一行的字符串的长度，那么读入n-1个字符，此时并没有读入\n因为并没有到行尾，同样在最后会插入'\0'.

example:

123abc

char s[5];

fgets(s,5,fp);

这时读入4个字符，123a,并没有换行符，所以strlen(s)=4.

fgets(...)读入整个文件内容

通常用while()循环来使fges()读入文本全部内容，并按行读入。

```
char s[1024];
 while((fgets(s,1024,fp))!=NULL)
 {
      printf(s);
 }
```

当然如果n小于每行的字符个数，也可以读，只不过读的次数要多。

假设一行为： 123456789

```
char s[2];
int  num=0;
while((fgets(s,2,fp))!=NULL)
 {
     printf(s);
     n++;
 }
```

每次读入一个字符， 最后也会读完一行，num=10，读了十次，所以，fgets若没遇到换行符，会接着从前一次的位置继续读入n-1个字符,只要是文本流没关闭。

读入空行的情况：

第一行 abcdef123

第二行

第三行 helloworld

其中第二行为空，fget(..)会把第二行也读入，因为并未到文件结尾。

有时我们并不需要空行，可以这样做。

```
while((fgets(s,n,fp))!=NULL)
  {
      if(strlen(s)!=1)   //注意这儿是1不是0，因为尽管是空行，它也会读入换行符，strlen(s)=1;
          printf(s);
  }
```

fgets(...)从标准设备读数据。

用fgets(...)还也读入标准输入设备(一般为键盘)的信息

原型 ： fgets(s,n,stdin);

假设在控制台下，我们可以用fgets(...)替代gets(),读入键盘输入的信息，fgets()是安全的，因为不会像gets()有溢出的可能。。

比如： 输入 abc

fgets(s,n,stdin)也会读入n-1个字符。但是只是从stdin流读入。。。