函数名: gets

功 能: 从流中取一字符串

用 法: char *gets(char *string);

程序例:

```
#include

int main(void)
{
char string[80];

printf("Input a string:");
gets(string);
printf("The string input was: %s\n",
string);
return 0;
}
```

gets()函数简单易用，它读取整行输入，直至遇到换行符，然后丢弃换行符，储存其余字符，并在这些字符的末尾添加一个空字符使其成为一个 C 字符串。它经常和 puts()函数配对使用，该函数用于显示字符串，并在末尾添加换行符。

缺点是 gets()唯一的参数是 words，它无法检查数组是否装得下输入行。上一章介绍过，数组名会被转换成该数组首元素的地址，因此，gets()函数只知道数组的开始处，并不知道数组中有多少个元素。如果输入的字符串过长，会导致缓冲区溢出（buffer overflow），即多余的字符超出了指定的目标空间。如果这些多余的字符只是占用了尚未使用的内存，就不会立即出现问题；如果它们擦写掉程序中的其他数据，会导致程序异常中止；或者还有其他情况。

函数名: fgets

功 能: 从流中读取一字符串

用 法: char *fgets(char *string, int n, FILE *stream);

程序例:

```
#include <string.h>
#include <stdio.h>

int main(void)
{
FILE *stream;
char string[] = "This is a test";
char msg[20];

/* open a file for update */
stream = fopen("DUMMY.FIL", "w+");

/* write a string into the file */
fwrite(string, strlen(string), 1, stream);

/* seek to the start of the file */
fseek(stream, 0, SEEK_SET);

/* read a string from the file */
fgets(msg, strlen(string)+1, stream);

/* display the string */
printf("%s", msg);

fclose(stream);
return 0;
}
```

fgets()函数通过第 2 个参数限制读入的字符数来解决溢出的问题。该函数专门设计用于处理文件输入，fgets()函数的第 2 个参数指明了读入字符的最大数量。如果该参数的值是 n，那么 fgets()将读入 n-1 个字符，或者读到遇到的第一个换行符为止。如果 fgets()读到一个换行符，会把它储存在字符串中。这点与 gets()不同，gets()会丢弃换行符。fgets()函数的第 3 个参数指明要读入的文件。如果读入从键盘输入的数据，则以 stdin（标准输入）作为参数，该标识符定义在 stdio.h 中。

fputs()函数返回指向 char 的指针。如果一切进行顺利，该函数返回的地址与传入的第 1 个参数相同。但是，如果函数读到文件结尾，它将返回一个特殊的指针：空指针（null pointer）。在代码中，可以用数字 0 来代替，不过在 C 语言中用宏 NULL 来代替更常见（如果在读入数据时出现某些错误，该函数也返回 NULL）。

函数名: puts

功　能: 送一字符串到流中

用　法: int puts(char *string);

程序例:

```
#include <stdio.h>
int main(void)
{
   char string[] = "This is an example output string\n";

   puts(string);
   return 0;
}
```

puts()在遇到空字符时就停止输出，同时在显示字符串时会自动在其末尾添加一个换行符。

函数名: fputs

功 能: 送一个字符到一个流中

用 法: int fputs(char *string, FILE *stream);

程序例:

```
#include <stdio.h>

int main(void)
{
/* write a string to standard output */
fputs("Hello world\n", stdout);

return 0;
}
```

fputs()函数的第 2 个参数指明要写入数据的文件。如果要打印在显示器上，可以用定义在 stdio.h 中的 stdout（标准输出）作为该参数。与 puts()不同，fputs()不会在输出的末尾添加换行符。

strlen

原型：extern int strlen(char *s);

用法：#include <string.h>

功能：计算字符串 s 的长度

说明：返回 s 的长度，不包括结束符 NULL。

举例：

```
// strlen.c

#include <syslib.h>
#include <string.h>

main()
{
  char *s="Golden Global View";

  clrscr();

  printf("%s has %d chars",s,strlen(s));

  getchar();
  return 0;
}
```

相关函数：无

strncat

原型：extern char *strncat(char *dest,char *src,int n);

用法：#include <string.h>

功能：把 src 所指字符串的前 n 个字符添加到 dest 结尾处(覆盖 dest 结尾处的'\0')并添加'\0'。

说明：src 和 dest 所指内存区域不可以重叠且 dest 必须有足够的空间来容纳 src 的字符串。

返回指向 dest 的指针。

举例：

```
// strncat.c

#include <syslib.h>
#include <string.h>

main()
{
  char d[20]="Golden Global";
  char *s=" View WinIDE Library";

  clrscr();

  strncat(d,s,5);
  printf("%s",d);

  getchar();
  return 0;
}
```

相关函数：strcat

strcmp

原型：extern int strcmp(char *s1,char * s2);

用法：#include <string.h>

功能：比较字符串 s1 和 s2。

说明：

当 s1<s2 时，返回值<0
当 s1=s2 时，返回值=0
当 s1>s2 时，返回值>0

举例：

```
// strcmp.c

#include <syslib.h>
#include <string.h>

main()
{
  char *s1="Hello, Programmers!";
  char *s2="Hello, programmers!";
  int r;

  clrscr();

  r=strcmp(s1,s2);
  if(!r)
    printf("s1 and s2 are identical");
  else
  if(r<0)
    printf("s1 less than s2");
  else
    printf("s1 greater than s2");


  getchar();
  return 0;
}
```

strncmp

原型：extern int strcmp(char *s1,char * s2, int n);
用法：#include <string.h>
功能：比较字符串 s1 和 s2 的前 n 个字符。
说明：
当 s1<s2 时，返回值<0
当 s1=s2 时，返回值=0
当 s1>s2 时，返回值>0

举例：

```
// strncmp.c
#include <syslib.h>
#include <string.h>
main()
{
  char *s1="Hello, Programmers!";
  char *s2="Hello, programmers!";
  int r;
  clrscr();
  r=strncmp(s1,s2,6);
  if(!r)
    printf("s1 and s2 are identical");
  else
  if(r<0)
    printf("s1 less than s2");
  else
    printf("s1 greater than s2");
  getchar();
  clrscr();
  r=strncmp(s1,s2,10);
  if(!r)
    printf("s1 and s2 are identical");
  else
  if(r<0)
    printf("s1 less than s2");
  else
    printf("s1 greater than s2");

  getchar();
  return 0;
}
```

stpcpy

原型：extern char *stpcpy(char *dest,char *src);

用法：#include <string.h>

功能：把 src 所指由 NULL 结束的字符串复制到 dest 所指的数组中。

说明：src 和 dest 所指内存区域不可以重叠且 dest 必须有足够的空间来容纳 src 的字符串。

返回指向 dest 结尾处字符(NULL)的指针。

举例：

```
// stpcpy.c

#include <syslib.h>
#include <string.h>

main()
{
  char *s="Golden Global View";
  char d[20];

  clrscr();

  stpcpy(d,s);
  printf("%s",d);

  getchar();
  return 0;
}
```

strncpy

原型：extern char *strncpy(char *dest, char *src, int n);

用法：#include <string.h>
功能：把 src 所指由 NULL 结束的字符串的前 n 个字节复制到 dest 所指的数组中。
说明：
如果 src 的前 n 个字节不含 NULL 字符，则结果不会以 NULL 字符结束。
如果 src 的长度小于 n 个字节，则以 NULL 填充 dest 直到复制完 n 个字节。
src 和 dest 所指内存区域不可以重叠且 dest 必须有足够的空间来容纳 src 的字符串。
返回指向 dest 的指针。
举例：

```
// strncpy.c

#include <syslib.h>
#include <string.h>

main()
{
  char *s="Golden Global View";
  char *d="Hello, GGV Programmers";
  char *p=strdup(s);

  clrscr();
  textmode(0x00);  // enable 6 lines mode

  strncpy(d,s,strlen(s));
  printf("%s\n",d);

  strncpy(p,s,strlen(d));
  printf("%s",p);


  getchar();
  return 0;
}
```

memccpy

原型：extern void *memccpy(void *dest, void *src, unsigned char ch, unsigned int count);

用法：#include <string.h>

功能:由 src 所指内存区域复制不多于 count 个字节到 dest 所指内存区域，如果遇到字符 ch 则停止复制。

说明：返回指向字符 ch 后的第一个字符的指针，如果 src 前 n 个字节中不存在 ch 则返回 NULL。ch 被复制。

举例：

```
// memccpy.c

#include <syslib.h>
#include <string.h>

main()
{
  char *s="Golden Global View";
  char d[20],*p;

  clrscr();

  p=memccpy(d,s,'x',strlen(s));
  if(p)
  {
    *p='\0';       // MUST Do This
    printf("Char found: %s.\n",d);
  }
  else
    printf("Char not found.\n");


  getchar();
  return 0;
}
```

memcpy

原型：extern void *memcpy(void *dest, void *src, unsigned int count);

用法：#include <string.h>

功能：由 src 所指内存区域复制 count 个字节到 dest 所指内存区域。

说明：src 和 dest 所指内存区域不能重叠，函数返回指向 dest 的指针。

举例：

```
// memcpy.c

#include <syslib.h>
#include <string.h>

main()
{
  char *s="Golden Global View";
  char d[20];

  clrscr();

  memcpy(d,s,strlen(s));
  d[strlen(s)]=0;
  printf("%s",d);

  getchar();
  return 0;
}
```