C语言常用的输入输出函数

函数原型如下：

```
#include <stdio.h>

int puts(const char *s);
char *gets(char *s);

int fputs(const char *s, FILE *stream);
char *fgets(char *s, int size, FILE *stream);

int putc(int c, FILE *stream);
int getc(FILE *stream);

int fputc(int c, FILE *stream);
int fgetc(FILE *stream);

int putchar(int c);
int getchar(void);
```

函数功能描述

puts()
输出一个字符串到标准输出流stdout，并在字符串结尾处自动添加换行符。

gets()
从标准输入设备读取一行字符保存到s指向的缓冲地址中，直到终端输入一个换行符或者读到EOF，并将换行符替换成一个空字符（'\0'）。它不会检查目标缓冲区是否能够容纳输入，所以很不安全。不要使用这个函数，应该使用fgets代替它。

fputs()
fputs()函数是puts的面向文件的版本，第二个参数用来说明输出的文件，可以使用stdout作为来调用，stdout在stdio.h中定义。与puts()不同，fputs()并不为输出自动添加换行符，也不会输出结尾的空字符。

fgets()
gets()函数有一个bug，它不检查预留存储区是否足够容纳输入的内容，fgets()改进了这个问题，它第二个参数让你指定最大读入字符数。同时fgets()会把读到的换行符存到字符串里。第三个参数用来指定读取哪一个文件，从键盘读数据时，可以使用stdin，这个标识符在stdio.h中定义。

fputc()
输出一个字符C到指定文件中，会将C转换成无符号字符类型。

fgetc()
从指定文件中读取下一个字符，返回字符的整数表示，或者是EOF，或者是错误。

putc()
putc()同等于fputc()函数，如果被实现成宏的形式，它的流不能是有副作用的表达式。

getc()
putc()同等于fputc()函数，如果被实现成宏的形式，它的流不能是有副作用的表达式。

putchar()
等同于putc(c, stdout)。

getchar()
等同于getc(stdin)。

从以上说明我们可以看出，我们不需要记住那么多的函数，只需要使用fgetc()，fputc()他fgets()和fputs()这2组基本可以满足开发需求。其他的要不就是这2组的宏封装，要不就是有Bug，可被替代。同时记住fgets()函数第二个参数用来指定读取的最大长度，输出超过这个长度或者遇到换行或者文件尾，就会返回。fgets()会保留读取的换行，fputs()不会自动添加换行。这样正好匹成一对。把文件指定为stdin或者stdout即可达到输出到标准io设备的效果。

```
int fputs(const char *s, FILE *stream);
char *fgets(char *s, int size, FILE *stream);

int fputc(int c, FILE *stream);
int fgetc(FILE *stream);
```

注意：
有副作用的表达式，指的是表达式执行后，会改变表达式中某些变量的值，例如(++, --)操作。这样的表达式不应该在宏里出现，否则在宏展开时候会出现意想之外的结果。例如计算平方的宏：

```
#include<stdio.h>

#define MACRO_SQRT(x) (x)*(x)
int func_sqrt(int x)
{
    return x * x;
}

int main(void)
{
    int x, y, n = 5, m = 5;
    x = MACRO_SQRT(++n);
    y = func_sqrt(++m);

    printf("MACRO_SQRT = %d\n", x); //49
    printf("func_sqrt  = %d\n", y); //36

    return 0;
}
```

宏与函数的区别在于，函数的参数是如果是表达式，会先计算出表达式的结果，再传递给函数使用。而宏是先展开，再计算表达式的值。宏展开后是这个样子的:

```
(++n) * (++n)
```

2次++后，n的值为7，结果便是7*7=49。为了避免这种问题，在给宏传递参数时，不要使用有副作用的表达式，即表达式中变量值会变化的表达式。

代码举例

```
#include<stdio.h>

int main(void)
{
    /*假设想输入1234567890这10个字符，
    * 多出的2个空间用来保存输入的换行符和自动添加的空字符(\0)
    */
    char input_str[12];

    fputs("input your name, please!\n", stdout);

    fgets(input_str, sizeof(input_str) / sizeof(char), stdin);

    fputs(input_str, stdout);

    return 0;
}
```

测试输出：

```
$ ./fputs
input your name, please!
1234567890
1234567890

$ ./fputs
input your name, please!
1234567890aaaa
1234567890a$
```

当多输出aaaa这几个字符时，输出最后的一个换行符被第一个a占据了，也就是说input_str数组中保存了'1234567890a\0'这12个字符。也就是说`char *fgets(char *s, int size, FILE *stream)`函数第二个参数如果指定为12，系统最多读取11个字符，并自动添加一个'\0'。

上面的程序有一个缺点，即fgets()函数会自动保存输入的换行，但是如果空间不够，换行就不会有，字符串里换行符(\n)的不确定性会给显示带来不方便，正如上例中那样。一个好的解决办法是，检测fgets()输入的字符串，去掉最后的换行符，在输出需要时，再考虑添加换行符。

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

int main(void)
{

    char *pos;

    char input_str[11];

    fputs("input your name, please!\n", stdout);

    fgets(input_str, sizeof(input_str) / sizeof(char), stdin);

    if (pos = strchr(input_str, '\n')) {
        *pos = '\0';
    }

    printf("%s\n", input_str);

    return 0;
}
```

上例展示了使用`strchr()`函数来定位输入的字符串最后是否有换行符，如果有，则把它去掉。