§13.输入输出流

要求：

1、安装UltraEdit软件，学会使用16进制方式查看文件，并掌握ASCII及16进制查看间的切换

2、完成本文档中所有的测试程序并填写运行结果，从而体会二进制与十进制文件在不同操作系统下的读写差异，掌握与文件有关的函数的正确用法（注意：本文档程序用C编译器）

3、需完成的页面，右上角有标注，直接在本文件上作答，用蓝色写出答案/截图即可

4、转换为pdf后提交

5、无特殊说明，Windows下用VS2017编译，Linux下用C++编译

6、因为篇幅问题，打开文件后均省略了是否打开成功的判断，这在实际应用中是不允许的

7、本题在“实验报告”中提交

1850059 计1班 杨志远

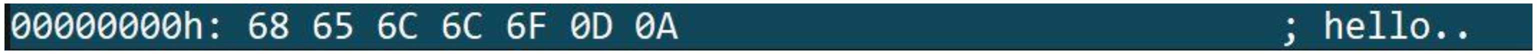
§13.输入输出流

本页需填写答案

例1：十进制方式写，在Windows/Linux下的差别

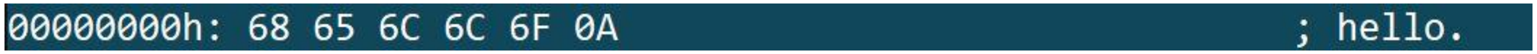
```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout;

    fout = fopen("out.txt", "w");
    fprintf(fout, "hello\n");
    fclose(fout);

    return 0;
}
```

Windows下运行，out.txt是_7_字节，用UltraEdit的16进制方式打开的贴图



```
00000000h: 68 65 6C 6C 6F 0D 0A                            ; hello..
```

Linux下运行，out.txt是_6_字节，用UltraEdit的16进制方式打开的贴图



```
00000000h: 68 65 6C 6C 6F 0A                               ; hello.
```

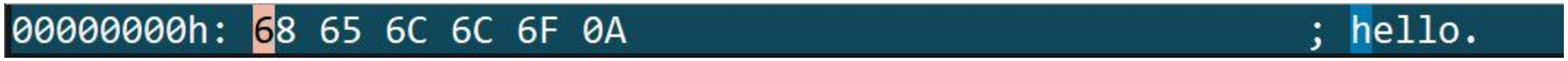
§13. 输入输出流

本页需填写答案

例2：二进制方式写，在Windows/Linux下的差别

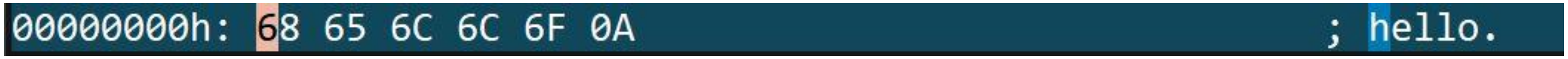
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout;

    fout = fopen("out.txt", "wb");
    fprintf(fout, "hello\n");
    fclose(fout);

    return 0;
}
```

Windows下运行，out.txt是_6_字节，用UltraEdit的16进制方式打开的贴图



```
00000000h: 68 65 6C 6C 6F 0A                                ; hello.
```

Linux下运行，out.txt是_6_字节，用UltraEdit的16进制方式打开的贴图



```
00000000h: 68 65 6C 6C 6F 0A                                ; hello.
```

§13.输入输出流

本页需填写答案

例3：十进制方式写，十进制方式读，0D0A在Windows下的表现

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "w");
    fprintf(fout, "hello\n");
    fclose(fout);

    fin = fopen("out.txt", "r");
    while (!feof(fin))
        printf("%d ", fgetc(fin));
    printf("\n");
    fclose(fin);
    return 0;
}
```

Windows下运行，输出结果是：

```
104 101 108 108 111 10 -1
```

说明：0D 0A在Windows的十进制方式下被当做_1_个字符处理，值是_10_。

§13.输入输出流

本页需填写答案

例4：十进制方式写，二进制方式读，0D0A在Windows下的表现

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "w");
    fprintf(fout, "hello\n");
    fclose(fout);

    fin = fopen("out.txt", "rb");
    while (!feof(fin))
        printf("%d ", fgetc(fin));
    printf("\n");
    fclose(fin);
    return 0;
}
```

Windows下运行，输出结果是：

```
104 101 108 108 111 13 10 -1
```

说明：0D 0A在Windows的二进制方式下被当做_2_个字符处理，值是_13和10_。

§13.输入输出流

本页需填写答案

例5：十进制方式写，十进制方式读，不同读方式在Windows下的表现

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

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "w");
    fprintf(fout, "hello\n");
    fclose(fout);

    char str[80];
    fin = fopen("out.txt", "r");
    fscanf(fin, "%s", str);
    printf("%d\n", strlen(str));
    printf("%d\n", fgetc(fin));
    fclose(fin);

    return 0;
}
```
	```
	#define _CRT_SECURE_NO_WARNINGS
	#include <stdio.h>
	#include <string.h>

	int main()
	{
	    FILE *fout, *fin;

	    fout = fopen("out.txt", "w");
	    fprintf(fout, "hello\n");
	    fclose(fout);

	    char str[80];
	    fin = fopen("out.txt", "r");
	    fgets(str, sizeof(str), fin); //课上未讲，自行查阅
	    printf("%d\n", strlen(str));
	    printf("%d\n", fgetc(fin));
	    fclose(fin);

	    return 0;
	}
	```

Windows下运行，输出结果是： 5 10 说明：fscanf()读到_0x0D_就结束了，_0x0A 还被留在缓冲区中，因此fgetc()读到了 _0x0A_。	Windows下运行，输出结果是： 6 -1 说明：fgets()读到_0x0A_就结束了，_0x0A_被读掉，因此fgetc()读到了_EOF_。

§13.输入输出流

本页需填写答案

例6：二进制方式写，十进制方式读，不同读方式在Windows下的表现

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

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "wb");
    fprintf(fout, "hello\n");
    fclose(fout);

    char str[80];
    fin = fopen("out.txt", "r");
    fscanf(fin, "%s", str);
    printf("%d\n", strlen(str));
    printf("%d\n", fgetc(fin));
    fclose(fin);

    return 0;
}
```
	```
	#define _CRT_SECURE_NO_WARNINGS
	#include <stdio.h>
	#include <string.h>

	int main()
	{
	    FILE *fout, *fin;

	    fout = fopen("out.txt", "wb");
	    fprintf(fout, "hello\n");
	    fclose(fout);

	    char str[80];
	    fin = fopen("out.txt", "r");
	    fgets(str, sizeof(str), fin); //课上未讲，自行查阅
	    printf("%d\n", strlen(str));
	    printf("%d\n", fgetc(fin));
	    fclose(fin);

	    return 0;
	}
	```

Windows下运行，输出结果是： 5 10 说明：fscanf()读到_0x6F_就结束了，_0x0A_还被留在缓冲区中，因此fgetc()读到了_0x0A_。	Windows下运行，输出结果是： 6 -1 说明：fgets()读到_0x0A_就结束了，_0x0A_被读掉，因此fgetc()读到了_EOF_。

§13. 输入输出流

本页需填写答案

例7：二进制方式写，二进制方式读，不同读方式在Windows下的表现

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

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "wb");
    fprintf(fout, "hello\n");
    fclose(fout);

    char str[80];
    fin = fopen("out.txt", "rb");
    fscanf(fin, "%s", str);
    printf("%d\n", strlen(str));
    printf("%d\n", fgetc(fin));
    fclose(fin);

    return 0;
}
```

Windows下运行，输出结果是：

5

10

说明：fscanf()读到_0x6F_就结束了，_0x0A_还被留在缓冲区中，因此fgetc()读到了_0x0A_。

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

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "wb");
    fprintf(fout, "hello\n");
    fclose(fout);

    char str[80];
    fin = fopen("out.txt", "rb");
    fgets(str, sizeof(str), fin); //课上未讲，自行查阅
    printf("%d\n", strlen(str));
    printf("%d\n", fgetc(fin));
    fclose(fin);

    return 0;
}
```

Windows下运行，输出结果是：

6

-1

说明：fgets()读到_0x0A_就结束了，_0x0A_被读掉，因此fgetc()读到了_EOF_。

§13.输入输出流

本页需填写答案

例8：十进制方式写，二进制方式读，不同读方式在Windows下的表现

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

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "w");
    fprintf(fout, "hello\n");
    fclose(fout);

    char str[80];
    fin = fopen("out.txt", "rb");
    fscanf(fin, "%s", str);
    printf("%d\n", strlen(str));
    printf("%d\n", fgetc(fin));
    fclose(fin);

    return 0;
}
```
	```
	#define _CRT_SECURE_NO_WARNINGS
	#include <stdio.h>
	#include <string.h>

	int main()
	{
	    FILE *fout, *fin;

	    fout = fopen("out.txt", "w");
	    fprintf(fout, "hello\n");
	    fclose(fout);

	    char str[80];
	    fin = fopen("out.txt", "rb");
	    fgets(str, sizeof(str), fin); //课上未讲，自行查阅
	    printf("%d\n", strlen(str));
	    printf("%d\n", fgetc(fin));
	    fclose(fin);

	    return 0;
	}
	```

Windows下运行，输出结果是： 5 13 说明：fscanf()读到_0x6F_就结束了，_0x0D_还被留在缓冲区中，因此fgetc()读到了_0x0D_。	Windows下运行，输出结果是： 7 -1 说明：fgets()读到_0x0A_就结束了，_0x0A_被读掉，因此fgetc()读到了_EOF_。

§13.输入输出流

本页需填写答案

例9：在Linux读取Windows下写的十进制文件

在Linux下运行本程序

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

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "wb");
    fprintf(fout, "hello\r\n"); //模拟Windows格式
    fclose(fout);

    char str[80];
    fin = fopen("out.txt", "rb");
    fscanf(fin, "%s", str);
    printf("%d\n", strlen(str));
    printf("%d\n", fgetc(fin));
    fclose(fin);

    return 0;
}
```

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

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "wb");
    fprintf(fout, "hello\n"); //比左侧少一个\r
    fclose(fout);

    char str[80];
    fin = fopen("out.txt", "rb");
    fscanf(fin, "%s", str);
    printf("%d\n", strlen(str));
    printf("%d\n", fgetc(fin));
    fclose(fin);

    return 0;
}
```

本例说明，在Linux下读取Windows格式的文件，要注意0D的处理

Linux下运行，输出结果是：

```
5
13
```

说明：fscanf读到_0x6F_就结束了，_0x0D_还被留在缓冲区中，因此fgetc()读到了_0x0D_。

Linux下运行，输出结果是：

```
5
10
```

说明：fscanf读到_0x6F_就结束了，_0x0A_还被留在缓冲区中，因此fgetc()读到了_0x0A_。

§13.输入输出流

本页需填写答案

例10：用十进制方式写入含\0的文件，观察文件长度

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout;

    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\0\x61\x62\x63");
    fclose(fout);

    return 0;
}
```

Windows下运行，out.txt的大小是_3_字节，Linux下运行，out.txt的大小是_3_字节

为什么？

fprintf在读取到字符串的尾零就就停止输出，\x61\x62\x63都被忽略。

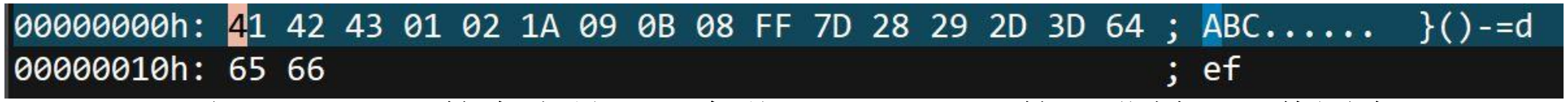
§13.输入输出流

本页需填写答案

例11：用十进制方式写入含非图形字符(ASCII码32是空格，33-126为图形字符)，但不含\0

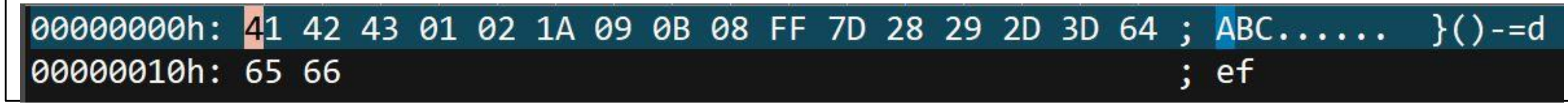
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout;

    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\x1\x2\x1A\t\v\b\xff\175()-=def");
    fclose(fout);

    return 0;
}
```

Windows下运行(VS有warning)，out.txt的大小是_18_字节，UltraEdit的16进制显示截图为：



```
00000000h: 41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64 ; ABC......  }()-=d
00000010h: 65 66                                           ; ef
```

Linux下运行，out.txt的大小是_18_字节，UltraEdit的16进制显示截图为：



```
00000000h: 41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64 ; ABC......  }()-=d
00000010h: 65 66                                           ; ef
```

§13.输入输出流

本页需填写答案

例12：用十进制方式写入含\x1A(十进制26=CTRL+Z)的文件，并用十进制/二进制方式读取

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\x1\x2\x1A\t\v\b\175()-=def");
    fclose(fout);

    fin = fopen("out.txt", "r");
    int c=0;
    while(!feof(fin)) {
        fgetc(fin);
        c++;
        }
    printf("c=%d\n", c);
    fclose(fin);

    return 0;
}
```

Windows下运行，文件大小：_17_
输出的c是：_6_
Linux下运行，文件大小：_17_
输出的c是：_18_
为什么？十进制方式下0x1A只在Windows上有效。

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\x1\x2\x1A\t\v\b\175()-=def");
    fclose(fout);

    fin = fopen("out.txt", "rb");
    int c=0;
    while(!feof(fin)) {
        fgetc(fin);
        c++;
        }
    printf("c=%d\n", c);
    fclose(fin);

    return 0;
}
```

Windows下运行，文件大小：_17_
输出的c是：_18_
Linux下运行，文件大小：_17_
输出的c是：_18_
c的大小比文件大小大_1_，原因是： fgetc读入字符时，由于feof()的滞后性，当读入EOF时，feof被置为true，在下次循环条件判断时退出，因此文件末尾的EOF也被计数了。二进制方式下0x1A在Windows上失效。

§13.输入输出流

本页需填写答案

例13：用十进制方式写入含\xFF(十进制255/-1，EOF的定义是-1)的文件，并用十/二进制读取

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\x1\x2\xFF\t\v\b\175()-=def");
    fclose(fout);

    fin = fopen("out.txt", "r");
    int c=0;
    while(!feof(fin)) {
        fgetc(fin);
        c++;
        }
    printf("c=%d\n", c);
    fclose(fin);

    return 0;
}
```

Windows下运行，文件大小：_17_
输出的c是：_18_
Linux下运行，文件大小：_17_
输出的c是：_18_

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\x1\x2\xFF\t\v\b\175()-=def");
    fclose(fout);

    fin = fopen("out.txt", "rb");
    int c=0;
    while(!feof(fin)) {
        fgetc(fin);
        c++;
        }
    printf("c=%d\n", c);
    fclose(fin);

    return 0;
}
```

Windows下运行，文件大小：_17_
输出的c是：_18_
Linux下运行，文件大小：_17_
输出的c是：_18_

综合例12~例13，结论：当文件中含字符_0x1A_时，不能用十进制方式读取，而当文件中含字符_0xFF_时，是可以用二/十进制方式正确读取的

§13.输入输出流

本页需填写答案

例14：比较格式化读和read()读的区别，并观察gcount()/tellg()在不同读入方式时值的差别

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABCDEFGHIJKLMNOPQRSTUVWXYZ\n");
    fclose(fout);

    fin = fopen("out.txt", "rb");
    char name[30];
    fscanf(fin, "%s", name);
    printf("*%s*\n", name);
    printf("%d\n", name[26]);
    printf("%d\n", ftell(fin));
    fclose(fin);

    return 0;
}
```

Windows下运行，文件大小：_28_
输出的name是：ABCDEFGHIJKLMNOPQRSTUVWXYZ
name[26]的值是：_0_
ftell的值是：_26_
说明：fscanf方式读入字符串时，和scanf方式相同，都是读到_分隔符_停止，并在数组最后加入一个_\0_。

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABCDEFGHIJKLMNOPQRSTUVWXYZ\n");
    fclose(fout);

    fin = fopen("out.txt", "rb");
    char name[30];
    fread(name, sizeof(name) / sizeof(name[0]),
sizeof(name[0]), fin);
    printf("*%s*\n", name);
    printf("%d\n", name[26]);
    printf("%d\n", ftell(fin));
    fclose(fin);

    return 0;
}
```

Windows下运行，文件大小：_28_
输出的name是：ABCDEFGHIJKLMNOPQRSTUVWXYZ
烫烫烫烫烫烫讥i
name[26]的值是：_13_
ftell的值是：_28_
说明：fread()读入时，是读到_EOF_停止，不在数组最后加入一个_\0_。

§13. 输入输出流

本页需填写答案

例15：比较read()读超/不超过文件长度时的区别，并观察gcount()/tellg()/good()的返回值

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout,"ABCDEFGHIJKLMNOPQRSTUVWXYZ");//无\n
    fclose(fout);

    fin = fopen("out.txt", "rb");
    char name[30] = "00000000000000000000000000000";
    fread(name, 20, 1, fin);
    printf("*%s*\n", name);
    printf("%d\n", name[20]);
    printf("%d\n", ftell(fin));
    printf("%d\n", feof(fin));
    fclose(fin);

    return 0;
}
```

Windows下运行，文件大小：_26_
输出的name是：ABCDEFGHIJKLMNOPQRST000000000
name[20]的值是：_48_
ftell()的值是：_20_
feof()的值是：_0_

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout,"ABCDEFGHIJKLMNOPQRSTUVWXYZ");//无\n
    fclose(fout);

    fin = fopen("out.txt", "rb");
    char name[30] = "00000000000000000000000000000";
    fread(name, 200, 1, fin);
    printf("*%s*\n", name);

    printf("%d\n", ftell(fin));
    printf("%d\n", feof(fin));
    fclose(fin);

    return 0;
}
```

Windows下运行，文件大小：_26_
输出的name是：ABCDEFGHIJKLMNOPQRSTUVWXYZ000

ftellg的值是：_26_
feof()的值是：_1_

§13.输入输出流

本页需填写答案

例16：使用seekg()移动文件指针，观察gcount()/tellg()/seekg()在不同情况下的返回值

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABCDEFGHIJKLMNOPQRSTUVWXYZ"); //无换行符
    fclose(fout);

    fin = fopen("out.txt", "rb");
    char name[80];
    fread(name, 10, 1, fin);
    printf("%d\n", ftell(fin));
    name[10] = '\0';
    printf("*%s*\n", name);
    fseek(fin, -5, SEEK_CUR);
    printf("%d\n", ftell(fin));
    fread(name, 10, 1, fin);
    printf("%d\n", ftell(fin));
    name[10] = '\0';
    printf("*%s*", name);
    fclose(fin);

    return 0;
}
```

Windows下运行，输出依次是：

```
10
*ABCDEFGHIJ*
5
15
*FGHIJKLMNO*
```

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABCDEFGHIJKLMNOPQRSTUVWXYZ"); //无换行符
    fclose(fout);

    fin = fopen("out.txt", "rb");
    char name[80];
    fread(name, 30, 1, fin);
    printf("%d\n", ftell(fin));
    name[30] = '\0';
    printf("*%s*\n", name);
    fseek(fin, 5, SEEK_SET);
    printf("%d\n", ftell(fin));
    fread(name, 30, 1, fin);
    printf("%d\n", ftell(fin));
    name[30] = '\0';
    printf("*%s*", name);
    fclose(fin);

    return 0;
}
```

Windows下运行，输出依次是：

```
26
*ABCDEFGHIJKLMNOPQRSTUVWXYZ烫烫*
5
26
*FGHIJKLMNOPQRSTUVWXYZVWXYZ烫烫*
```

§13.输入输出流

本页需填写答案

例17：freopen的使用

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp;

    fp = freopen("out.txt", "w",stdout);
    printf("Hello, world!\n");
    fclose(fp);

    return 0;
}
```
	```
	#define _CRT_SECURE_NO_WARNINGS
	#include <stdio.h>

	int main()
	{
	    FILE *fp;
	    int a, b;

	    fp = freopen("in.txt", "r", stdin);
	    scanf("%d %d", &a, &b);
	    printf("a=%d b=%d\n", a, b);
	    fclose(fp);

	    return 0;
	}
	```

Windows下运行后，屏幕输出：_没有输出_ out.txt中内容：_Hello, world!_ 思考：删除out.txt文件，将打开方式从"w"换为"r"，运行结果？为什么？ 运行时错误 Debug Assertion Failed 程序将标准输出流重定向为只读文件，会出现错误	准备工作：在当前目录下建in.txt文件写入两个整数 Windows下运行，输出是：_a=4 b=7_ 思考：删除in.txt后运行，运行结果？为什么？ 运行时错误 Debug Assertion Failed 没有找到文件in.txt，fp的值为NULL，标准输入流重定向到一个空指针对应的文件，会出现错误

§13.输入输出流

本页需填写答案

例18：fread的返回值理解

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp;
    char buf[80];

    fp = fopen("in.txt", "r");
    int ret = fread(buf, 26, 1, fp);
    printf("ret=%d\n", ret);
    fclose(fp);

    return 0;
}
```

准备工作：在当前目录下建in.txt文件，
　　　　　写入A..Z共26个字母

fread的第2/3参数：

原26,1，ret=1
换1,26，ret=26
换13,2，ret=2
换2,13，ret=13
换80,1，ret=0
换1,80，ret=26
换15,2，ret=1
换2,15，ret=13

§13. 输入输出流

本页需填写答案

例19：fwrite返回值理解

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp;
    char buf[80]="abcdefghijklmnopqrstuvwxyz";

    fp = fopen("out.txt", "w");
    int ret = fwrite(buf, 26, 1, fp);
    printf("ret=%d\n", ret);
    fclose(fp);

    return 0;
}
```

fwrite的第2/3参数：

原26,1，ret=1，文件大小26
换1,26，ret=26，文件大小26
换13,2，ret=2，文件大小26
换2,13，ret=13，文件大小26
换80,1，ret=1，文件大小80
换1,80，ret=80，文件大小80
换15,2，ret=2，文件大小30
换2,15，ret=15，文件大小30