

## § 13. 输入输出流

要求:

- 1、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的读写差异
- 2、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**即可
- 3、转换为pdf后提交
- 4、无特殊说明，Windows下用VS2017编译，Linux下用C++编译

〈朱世轩 计2 1752528〉

## § 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

```
#include<iostream>
#include<fstream>
using namespace std;

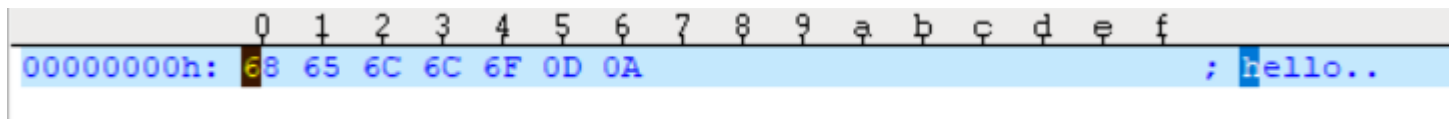
int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);

    out << "hello" << endl;

    out.close();

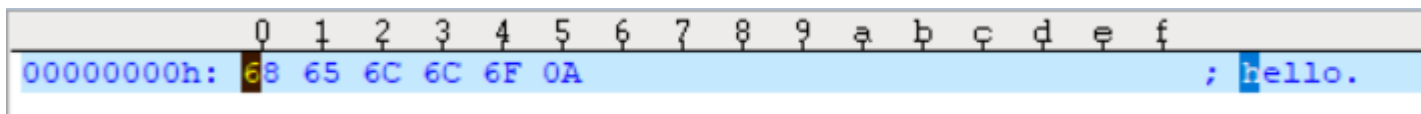
    return 0;
}
```

Windows下运行，out.txt是\_\_7\_\_字节，用UltraEdit的16进制方式打开的贴图



0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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



0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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

```
#include<iostream>
#include<fstream>
using namespace std;

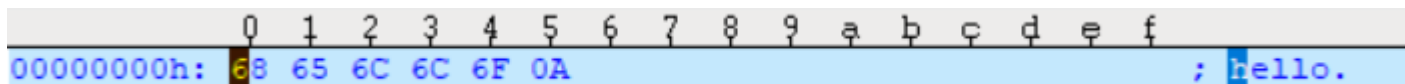
int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);

    out << "hello" << endl;

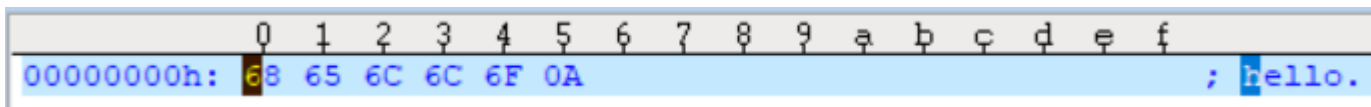
    out.close();

    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



Linux下运行，out.txt是 6 字节，用UltraEdit的16进制方式打开的贴图



## § 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

```
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    ifstream in("out.txt", ios::in);
    while(!in.eof())
        cout << in.get() << ' ';
    cout << endl;
    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\_\_0A\_\_。

## § 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

```
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    while(!in.eof())
        cout << in.get() << ' ';
    cout << endl;
    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\_\_0D 0A\_\_。

## § 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

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in >> str;
    cout << strlen(str) << endl;
    cout << in.get() << endl;

    return 0;
}
```

Windows下运行，输出结果是：

5

10

说明：in>>str读到0D就结束了，0A还被留在缓冲区中，因此in.get()读到了0A。

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;

    return 0;
}
```

Windows下运行，输出结果是：

5

-1

说明：in.getline读到0D就结束了0D 0A被读掉，因此in.get()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

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in >> str;
    cout << strlen(str) << endl;
    cout << in.get() << endl;

    return 0;
}
```

Windows下运行，输出结果是：

5

10

说明：in>>str读到\_0D\_\_就结束了，\_0A\_\_还被留在缓冲区中，因此in.get()读到了\_0A\_\_。

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;

    return 0;
}
```

Windows下运行，输出结果是：

5

-1

说明：in.getline读到\_0D\_就结束了，0D 0A被读掉，因此in.get()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

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in >> str;
    cout << strlen(str) << endl;
    cout << in.get() << endl;

    return 0;
}
```

Windows下运行，输出结果是：

5

10

说明：in>>str读到\_\_0D\_\_就结束了，\_\_0A\_\_  
还被留在缓冲区中，因此in.get()读到了  
\_\_0A\_\_。

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;

    return 0;
}
```

Windows下运行，输出结果是：

5

-1

说明：in.getline读到\_\_0D\_\_就结束了，0D 0A  
被读掉，因此in.get()读到了\_\_0A\_\_。



## § 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，不同方式在Windows下的表现

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in >> str;
    cout << strlen(str) << endl;
    cout << in.get() << endl;

    return 0;
}
```

Windows下运行，输出结果是：

5

13

说明：in>>str读到\_\_OD\_\_就结束了，\_OD\_还被留在缓冲区中，因此in.get()读到\_OD\_\_。

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;

    return 0;
}
```

Windows下运行，输出结果是：

6

-1

说明：1、in.getline读到\_\_OD\_\_就结束了，\_\_OD\_OA\_被读掉，因此in.peek()读到\_EOF\_。  
2、strlen(str)是\_6\_，最后一个字符是\_\0\_

## § 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

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
```

在Linux下运行本程序

```
int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello\r" << endl; //模拟Windows格式
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;

    return 0;
}
```

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
```

同例8右侧，未变过

```
int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;

    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：

6

-1

说明：1、in.getline读到\_\_0D\_\_就结束了，  
0D 0A被读掉，因此in.peek()读到了\_EOF\_。

2、strlen(str)是\_6\_，最后一个字符是\_\0\_

Windows下运行，输出结果是：

6

-1

说明：1、in.getline读到\_0D\_就结束了0D 0A  
被读掉，因此in.get()读到了\_EOF\_。

2、strlen(str)是\_6\_，最后一个字符是\_\0\_

## § 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

```
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\0\x61\x62\x63" << endl;
    out.close();

    return 0;
}
```

Windows下运行，out.txt的大小是\_\_5\_\_字节，Linux下运行，out.txt的大小是\_\_4\_\_字节

为什么？

ABC为3字节，读到\0结束，windows中endl为\r\n占2字节，而linux为\n占1字节。所以windows下为5字节，linux下为4字节

## § 13. 输入输出流

本页需填写答案

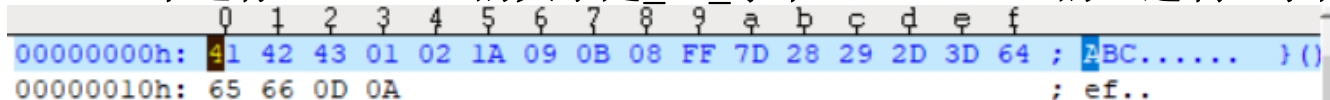
例11：用十进制方式写入含非图形字符(ASCII码32是空格，33-126为图形字符)，但不含\0

```
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\xff\175()-=def" << endl;
    out.close();

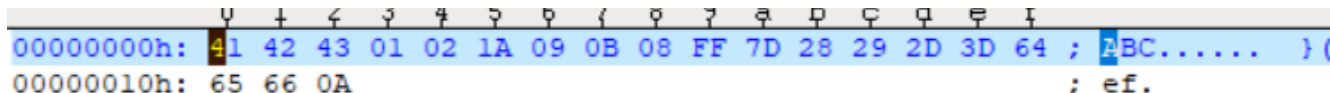
    return 0;
}
```

Windows下运行，out.txt的大小是\_20\_字节，UltraEdit的16进制显示截图为：



0 1 2 3 4 5 6 7 8 9 a b c d e f  
00000000h: 41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64 ; ABC..... }()  
00000010h: 65 66 0D 0A ; ef..

Linux下运行，out.txt的大小是\_19\_字节，UltraEdit的16进制显示截图为：



0 1 2 3 4 5 6 7 8 9 a b c d e f  
00000000h: 41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64 ; ABC..... }()  
00000010h: 65 66 0A ; 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

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\xff\175()--def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in);
    int c=0;
    while(!in.eof()) {
        in.get();
        c++;
    }
    cout << c << endl;

    return 0;
}
```

Windows下运行，文件大小： 20  
输出的c是： 6  
Linux下运行，文件大小： 19  
输出的c是： 20

为什么？windows下读到\x1A直接结束，而linux下文件结束标志不为ctrl+z，正常读取

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\xff\175()--def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    int c=0;
    while(!in.eof()) {
        in.get();
        c++;
    }
    cout << c << endl;

    return 0;
}
```

Windows下运行，文件大小： 20  
输出的c是： 21  
Linux下运行，文件大小： 19  
输出的c是： 20

c的大小比文件大小大\_1\_，原因是：遇到EOF使c多累加了一次，而EOF不计入文件大小

## § 13. 输入输出流

本页需填写答案

例13：用十进制方式写入含\x1A(十进制26=CTRL+Z)的文件，并用十进制不同方式读取

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\175()--def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in); //不加ios::binary
    int c=0;
    while(in.get() != EOF) {
        c++;
    }
    cout << c << endl;

    return 0;
}
```

Windows下运行，文件大小： 19  
输出的c是： 5  
Linux下运行，文件大小： 18  
输出的c是： 18

为什么？ windows下读到\x1A直接结束，而  
linux下文件结束标志不为ctrl+z，正常读取

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\175()--def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in); //不加ios::binary
    int c=0;
    char ch;
    while((ch=in.get()) != EOF) {
        c++;
    }
    cout << c << endl;

    return 0;
}
```

Windows下运行，文件大小： 19  
输出的c是： 5  
Linux下运行，文件大小： 18  
输出的c是： 18

为什么？ windows下读到\x1A直接结束，而  
linux下文件结束标志不为ctrl+z，正常读取。  
判断了读到EOF结束，所以c与文件大小相同

## § 13. 输入输出流

本页需填写答案

例14：用十进制方式写入含\xFF(十进制255/-1)的文件，并进行正确/错误读取

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\xff\t\v\b\175()--def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in); //可加ios::binary
    int c=0;
    while(in.get() != EOF) {
        c++;
    }
    cout << c << endl;

    return 0;
}
```

Windows下运行，文件大小： 19字节

输出的c是： 18

Linux下运行，文件大小： 18字节

输出的c是： 18

为什么？正确读取时\xFF作为正常字符处理，可以正常读取

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\xff\t\v\b\175()--def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in); //可加ios::binary
    int c=0;
    char ch;
    while((ch=in.get()) != EOF) {
        c++;
    }
    cout << c << endl;

    return 0;
}
```

Windows下运行，文件大小： 19字节

输出的c是： 5

Linux下运行，文件大小： 18字节

输出的c是： 5

为什么？将\xFF赋值给字符后作为EOF处理，读到EOF时c中止累加

综合例12~例14，结论：当文件中含字符\x1A\_时，不能用十进制方式读取，而当文件中含字符\_\xFF\_时，是可以用二/十进制方式正确读取的（emmmm...，世上本无事，你偏要找点事出来）