实验1.2

1. 实验内容

1.将文件内容复制到另一个目标文件。这个程序首先提示用户输入源文件和目标文件的名称。利用Windows或POSIX的API，编写这个程序。确保包括所有必要的错误检查以及源文件存在。

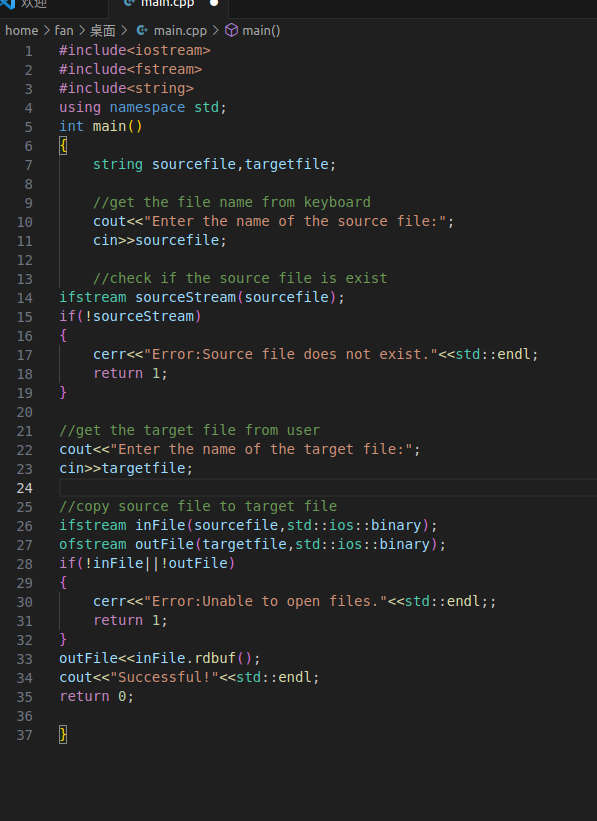
2.在正确设计并测试这个程序后，指出系统调用部分。

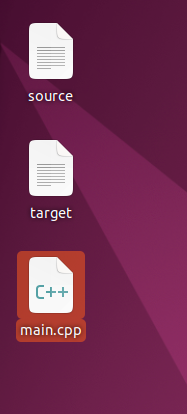
二．实验目的

1.通过实验，加深对系统调用的理解。

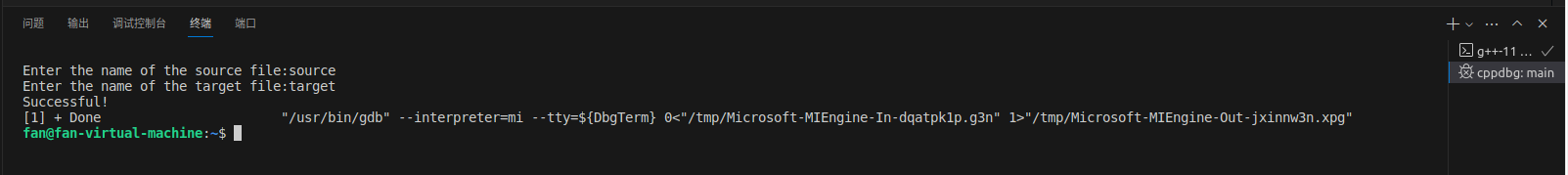
2.通过编写简单的程序，加深对Linux操作系统的理解，并且能初步掌握 C/C++编译和调试工具。

1. 设计思路
2. 首先，创建变量以存放源文件和目标文件。
3. 给予用户提示，输入源文件名称。
4. 设置异常处理，判断源文件是否存在，如果源文件无法打开，则抛出异常。
5. 设置另一个变量，用户输入目标文件。
6. 抛出异常，如果源文件或目标文件无法打开，则抛出异常。
7. 若没有异常，通过流操作读出源文件到目标文件。
8. 源代码





1. 程序运行结果



1. 总结
2. 使用系统调用部分
3. 打开文件（open）：

函数原型：#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int open(const char \*pathname, int flags, mode\_t mode);

pathname:要打开的文件路径,可以是绝对路径或者相对路径。

flags:打开文件时的操作标志,可以使用按位或运算组合。

mode:如果flags中指定了O\_CREAT,则mode决定了新建文件的访问权限模式。

返回值:成功返回新打开的文件描述符,失败返回-1,具体错误原因可以通过errno获得。

1. 读取文件（read）：

函数原型：ssize\_t read(int fd, void \*buf, size\_t count);

fd:要读取数据的文件描述符,一般是open()或其他 打开函 数返回的文件描述符。

buf:读取数据存放的缓冲区指针。

count:要读取的字节数。

read()的工作过程是:

从文件描述符fd所指向的打开文件中读取最多count个字 节的数据，将读取到的数据写入buf指向的缓冲区。返回实际读取的字节数。如果文件末尾已经到达返回0,错误返回-1。

1. 写入文件（write）：

函数原型：ssize\_t write(int fd, const void \*buf, size\_t count);

fd:要写入数据的文件描述符,一般是open()或

其他打开函数返回的文件描述符。

buf:待写入数据的缓冲区指针。

count:要写入的数据字节数。

write()的工作过程是：从buf指向的缓冲区读取最多count个 字节的数据，将读取的数据写入到文件描述符fd所指向的打开文件中。返回实际写入的字节数。如果发生错误返回-1。

1. 关闭文件（close）：

函数原型：int close(int fd);

fd:要关闭的文件描述符,一般是open()/read()/write() 等函数打开文件后返回的文件描述符。

close()调用后会关闭与文件描述符fd关联的文件访问权限。释 放系统用来跟踪该文件描述符的资源,如内核数据结构等。fd后续不再有效,不允许再通过fd访问文件。