**广州大学学生实验报告**

**开课学院及实验室：**计算机科学与网络工程学院软件实验室 **2023年 5月 日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | **计算机科学与网络工程学院** | **年级/专业/班** | **软件211** | **姓名** | 张景致 | **学号** | 32106300004 |
| **实验课程名称** | 操作系统实验 | | | | | **成绩** |  |
| **实验项目名称** | **文件系统** | | | | | **指导老师** |  |

**实验四 文件系统**

1. **实验目的**
2. 熟悉Linux文件系统的文件和目录结构，掌握Linux文件系统的基本特征；
3. 模拟实现Linux文件系统的简单I/O流操作：备份文件。
4. **实验内容**
5. 浏览Linux系统根目录下的子目录，熟悉每个目录的文件和功能；
6. 设计程序模拟实现Linux文件系统的简单I/O流操作：备份文件。
7. **实验原理**
8. Linux各种发行版的目录结构基本一致，各个目录简单介绍如下：

|  |  |
| --- | --- |
| **目录** | **描述** |
| / | 根目录 |
| /bin | 做为基础系统所需要的最基础的命令就是放在这里。比如 ls、cp、mkdir等命令；功能和/usr/bin类似，这个目录中的文件都是可执行的，普通用户都可以使用的命令。 |
| /boot | Linux的内核及引导系统程序所需要的文件，比如 vmlinuz initrd.img 文件都位于这个目录中。在一般情况下，GRUB或LILO系统引导管理器也位于这个目录；启动装载文件存放位置，如kernels,initrd,grub。一般是一个独立的分区。 |
| /dev | 一些必要的设备,声卡、磁盘等。还有如 /dev/null. /dev/console /dev/zero /dev/full 等。 |
| /etc | 系统的配置文件存放地. 一些服务器的配置文件也在这里；比如用户帐号及密码配置文件；  /etc/opt:/opt对应的配置文件  /etc/X11:Xwindows系统配置文件  /etc/xml:XML配置文件  …… |
| /home | 用户工作目录，和个人配置文件，如个人环境变量等，所有的账号分配一个工作目录。一般是一个独立的分区。 |
| /lib | 库文件存放地。bin和sbin需要的库文件。类似windows的DLL。 |
| /media | 可拆卸的媒介挂载点，如CD-ROMs、移动硬盘、U盘，系统默认会挂载到这里来。 |
| /mnt | 临时挂载文件系统。这个目录一般是用于存放挂载储存设备的挂载目录的，比如有cdrom 等目录。可以参看/etc/fstab的定义。 |
| /opt | 可选的应用程序包。 |
| /proc | 操作系统运行时，进程（正在运行中的程序）信息及内核信息（比如cpu、硬盘分区、内存信息等）存放在这里。/proc目录伪装的文件系统proc的挂载目录，proc并不是真正的文件系统，它的定义可以参见 /etc/fstab 。 |
| /root | Root用户的工作目录 |
| /sbin | 和bin类似，是一些可执行文件，不过不是所有用户都需要的，一般是系统管理所需要使用得到的。 |
| /tmp | 系统的临时文件，一般系统重启不会被保存。 |
| /usr | 包含了系统用户工具和程序。  /usr/bin：非必须的普通用户可执行命令  /usr/include：标准头文件   /usr/lib:/usr/bin/ 和 /usr/sbin/的库文件   /usr/sbin:非必须的可执行文件  /usr/src:内核源码  /usr/X11R6:X Window System, Version 11, Release 6. |
| /srv | 该目录存放一些服务启动之后需要提取的数据 |

1. Linux的文件结构是单个的树状结构.可以用tree进行展示。文件操作命令见附录1，
2. Linux文件系统：Linux 中允许众多不同的文件系统共存，如 ext2, ext3, vfat 等。通过使用同一套文件 I/O 系统 调用即可对 Linux 中的任意文件进行操作而无需考虑其所在的具体文件系统格式；更进一步，对文件的 操作可以跨文件系统而执行。“一切皆是文件”是 Unix/Linux 的基本哲学之一。不仅普通的文件，目录、字符设备、块设备、 套接字等在 Unix/Linux 中都是以文件被对待；它们虽然类型不同，但是对其提供的却是同一套操作界面。
3. 虚拟文件系统（Virtual File System, 简称 VFS）， 是 Linux 内核中的一个软件层，用于给用户空间的程序提供文件系统接口；同时，它也提供了内核中的一个 抽象功能，允许不同的文件系统共存。系统中所有的文件系统不但依赖 VFS 共存，而且也依靠 VFS 协同工作。为了能够支持各种实际文件系统，VFS 定义了所有文件系统都支持的基本的、概念上的接口和数据 结构；同时实际文件系统也提供 VFS 所期望的抽象接口和数据结构，将自身的诸如文件、目录等概念在形式 上与VFS的定义保持一致。换句话说，一个实际的文件系统想要被 Linux 支持，就必须提供一个符合VFS标准 的接口，才能与 VFS 协同工作。实际文件系统在统一的接口和数据结构下隐藏了具体的实现细节，所以在VFS 层和内核的其他部分看来，所有文件系统都是相同的。

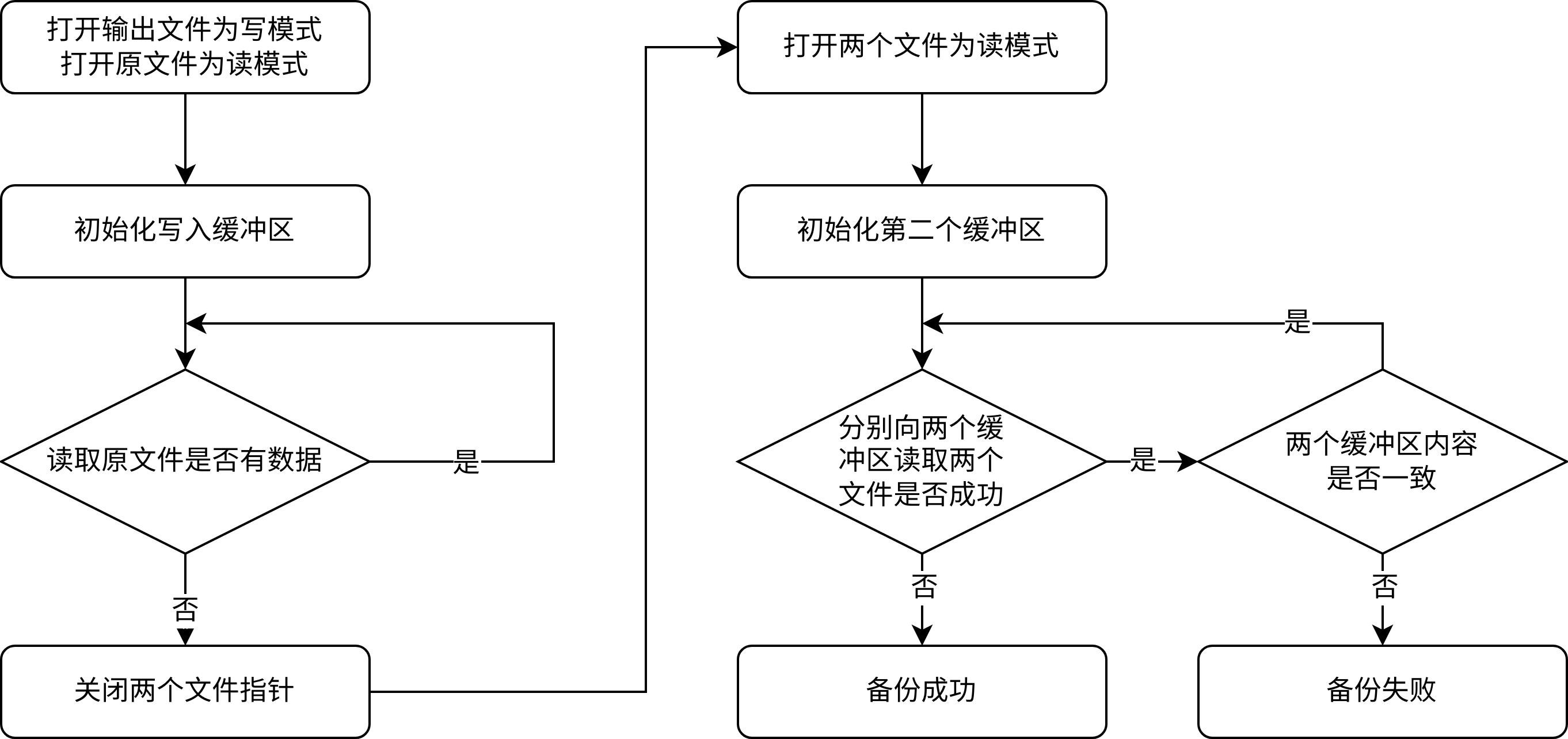
5、利用C库函数fopen(), fread(), fwrite(), fclose() 来实现简单的文件备份， 即将一个文件的内容拷贝到另一个文件中去。

1. **实验中用到的系统调用函数**

实验只是模拟实现文件的备份功能，不需要系统调用函数。

**五、实验正文**

1. 对文件系统命令和库函数要提前熟悉；
2. 画出备份文件的过程图；



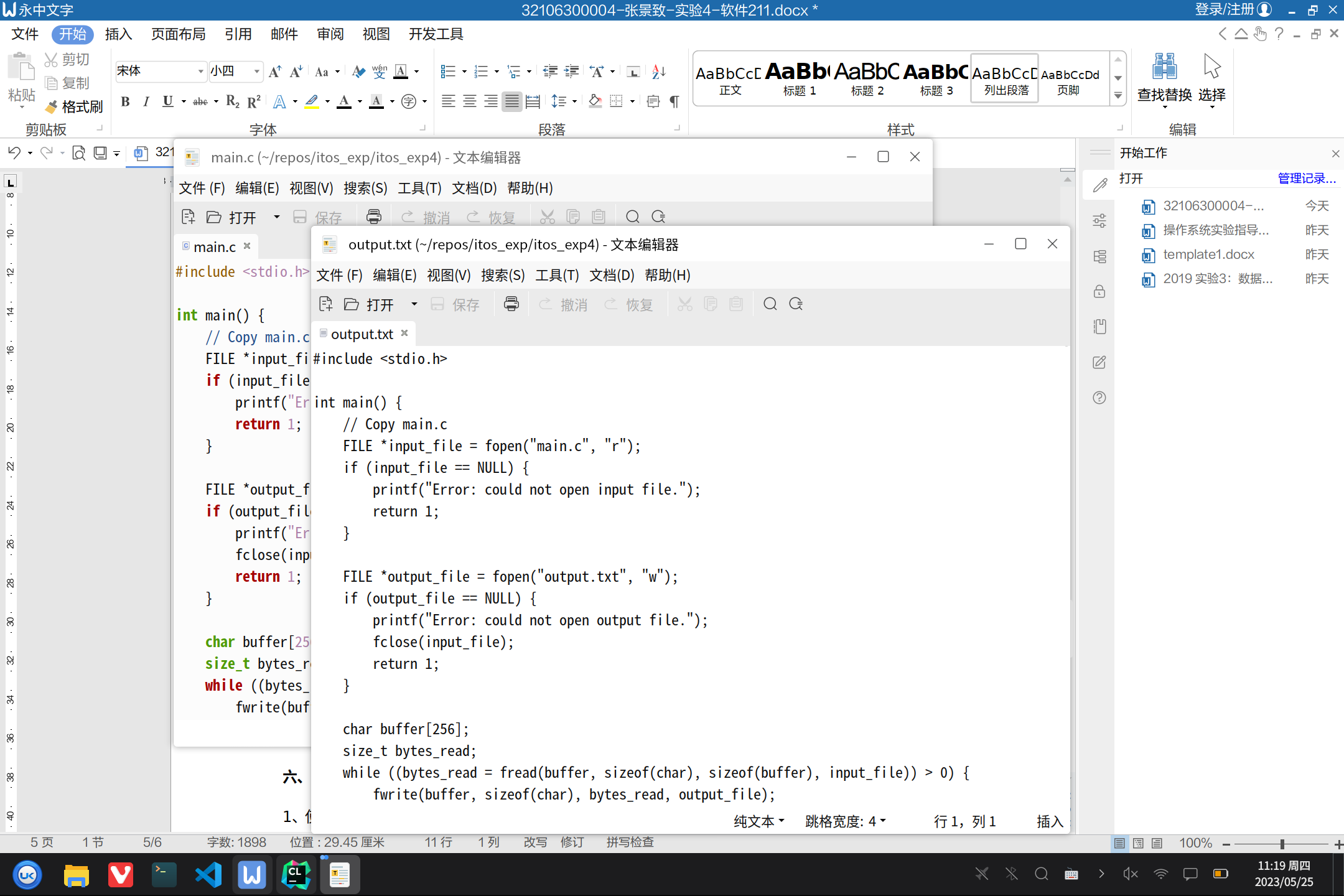
3、 测试数据文件提前准备好；

1. 编写程序并调试；

main.c

#include <stdio.h>  
  
int main() {  
 *// Copy main.c* FILE \*input\_file = fopen("main.c", "r");  
 if (input\_file == **NULL**) {  
 printf("Error: could not open input file.");  
 return 1;  
 }  
  
 FILE \*output\_file = fopen("output.txt", "w");  
 if (output\_file == **NULL**) {  
 printf("Error: could not open output file.");  
 fclose(input\_file);  
 return 1;  
 }  
  
 char buffer[256];  
 size\_t bytes\_read;  
 while ((bytes\_read = fread(buffer, sizeof(char), sizeof(buffer), input\_file)) > 0) {  
 fwrite(buffer, sizeof(char), bytes\_read, output\_file);  
 }  
  
 fclose(input\_file);  
 fclose(output\_file);  
  
 *// compare 2 files (main.c and output.txt)* char buffer\_b[256];  
 FILE \*input\_file\_b = fopen("main.c", "r");  
 FILE \*output\_file\_b = fopen("output.txt", "r");  
  
 while (fread(buffer, sizeof(char), sizeof(buffer), input\_file\_b) > 0) {  
 fread(buffer\_b, sizeof(char), sizeof(buffer\_b), output\_file\_b);  
 for (int i = 0; i < 256; i++) {  
 if (buffer[i] != buffer\_b[i]) {  
 printf("Error: files do not match.");  
 fclose(input\_file\_b);  
 fclose(output\_file\_b);  
 return 1;  
 }  
 }  
 }  
  
 printf("Files match.");  
 fclose(input\_file\_b);  
 fclose(output\_file\_b);  
  
 return 0;  
}

1. 截屏输出实验结果；



**六、思考题**

1. 使用系统调用函数open(),read(),write(),close()实现简单文件备份的原理是什么？

open 创建一个新的文件描述符，如果调用成功，返回一个可以被其他IO系统调用使用的FD。

read从FD指向的文件里读入 nbytes 个字节的数据，写入缓冲 buf 中。

write 把缓冲 buf 的前 nbytes 个字节写入与FD所指向的文件中，返回实际写入的字节数。

close 关闭 FD，使其不能在被使用。

2、使用C库函数fopen(), fread(), fwrite(), fclose() 来实现简单文件备份的原理是什么？

（一）C库函数的作法是将要处理的数据先存入缓冲区内，等到缓冲区装满了，再将数据一次写入或者读出。这种方式处理小量数据时效率比较高。  
（二）库函数对文件的操作实际上是通过以上的系统调用来实现的。例如C库函数fwrite()就是通过write()系统调用来实现的。

3、上述二者的区别在哪里？

没有F前缀的版本主要存在于POSIX C标准库中，适用于（类）Unix平台；有F前缀的版本主要存在于ISO C标准库中，几乎适用于所有平台。

（一）系统调用函数：在用户态调用，在内核态执行；C库函数：在用户态调用，在用户态执行。  
（二）各个系统的系统调用是不同的；C函数库是相同的。  
（三）系统调用需要在切换到内核上下文环境再切换回来，开销比较大；而C库函数属于过程调用，开销比较小。