

**HUBEI UNIVERSITY OF AUTOMOTIVE TECHNOLOGY**



**操作系统原理**

**实 验 报 告**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验项目 | 文件操作实验 | | | |
| 学生姓名 | 章崇文 | 学生学号 | 202202296 | |
| 学生班级 | 计算机222 | 完成日期 | 2024.6.5 | |
| 实验成绩 |  | 评阅日期 | |  |
| 评阅教师 |  | | | |

**湖北汽车工业学院实验报告**

班 号 学 号   姓 名

选课班中的序号  完成日期 年 月 日 至 节

# 实验六 文件操作实验

一、实验目的

1、了解文件系统的功能及实现原理。

2、掌握LINUX下文件操作的有关系统调用。

3、通过模拟程序实现简单的一级文件系统或二级文件系统。

**二、实验内容**

1、编程显示文件自身。（1分）

运行结果截屏：（包含编译、运行命令）

1. 编程实现文件加密和解密。（4分）
2. 编程实现文件的合并（尝试用不同方法实现）。（3分）
3. **编程实现一个简单的文件系统，要求实现以下功能：**

文件创建 文件解密 文件显示 文件列目录 文件删除

文件加密 文件合并 文件查询 文件复制 文件重命名

**要求列出程序编译、运行过程以及执行结果的部分截屏。（附加题）**

**注：选做该题的同学，不用做前面的题，直接提交这一道题的程序和结果即可。**

**代码：**

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <filesystem> // 需要C++17支持，用于文件系统操作

class FileSystem

{

public:

// 文件创建

bool createFile(const std::string &filename)

{

std::ofstream file(filename);

if (file.is\_open())

{

file.close();

return true;

}

return false;

}

// 文件显示（简单读取并打印到控制台）

void displayFile(const std::string &filename)

{

std::ifstream file(filename);

if (file.is\_open())

{

std::string line;

while (std::getline(file, line))

{

std::cout << line << std::endl;

}

file.close();

}

else

{

std::cerr << "无法打开文件: " << filename << std::endl;

}

}

// 文件列目录

void listDirectory(const std::string &directory)

{

for (const auto &entry : std::filesystem::directory\_iterator(directory))

{

std::cout << entry.path() << std::endl;

}

}

// 文件删除

bool deleteFile(const std::string &filename)

{

if (std::filesystem::exists(filename))

{

try

{

std::filesystem::remove(filename);

return true;

}

catch (std::filesystem::filesystem\_error &e)

{

std::cerr << "删除文件时出错：" << e.what() << std::endl;

}

}

else

{

std::cerr << "文件不存在: " << filename << std::endl;

}

return false;

}

bool encryptFile(const std::string &inputFile, const std::string &outputFile)

{

const char key = 0x7F; // 示例密钥，可以替换为任意字符

std::ifstream inFile(inputFile, std::ios::binary);

std::ofstream outFile(outputFile, std::ios::binary);

if (!inFile.is\_open() || !outFile.is\_open())

{

std::cerr << "无法打开文件" << std::endl;

return false;

}

char ch;

while (inFile.get(ch))

{ // 逐字节读取

// 对字符进行XOR加密

char encryptedCh = ch ^ key;

outFile.put(encryptedCh); // 写入加密后的字符

}

inFile.close();

outFile.close();

return true; // 如果一切顺利，返回true

}

bool decryptFile(const std::string &inputFile, const std::string &outputFile)

{

// 使用与加密时相同的密钥

const char key = 0x7F;

std::ifstream inFile(inputFile, std::ios::binary);

std::ofstream outFile(outputFile, std::ios::binary);

if (!inFile.is\_open() || !outFile.is\_open())

{

std::cerr << "无法打开文件" << std::endl;

return false;

}

char ch;

while (inFile.get(ch))

{

// 对字符进行XOR解密，注意加密和解密使用的操作是相同的（XOR的逆运算是其本身）

char decryptedCh = ch ^ key;

outFile.put(decryptedCh);

}

inFile.close();

outFile.close();

return true; // 解密成功

}

// 文件合并函数

bool mergeFiles(const std::vector<std::string> &files, const std::string &outputFile)

{

std::ofstream outFile(outputFile, std::ios::binary | std::ios::app); // 打开或创建输出文件，并设置为追加模式

if (!outFile.is\_open())

{

std::cerr << "无法打开输出文件: " << outputFile << std::endl;

return false;

}

for (const auto &file : files)

{

std::ifstream inFile(file, std::ios::binary);

if (!inFile.is\_open())

{

std::cerr << "无法打开输入文件: " << file << std::endl;

outFile.close(); // 记得关闭已打开的输出文件

return false;

}

// 将当前文件内容追加到输出文件

outFile << inFile.rdbuf(); // 使用rdbuf()效率较高，直接拷贝缓冲区

inFile.close(); // 关闭当前输入文件

}

outFile.close(); // 最后关闭输出文件

return true; // 如果一切顺利，返回true

}

void searchFile(const std::string &directory, const std::string &filename)

{

for (const auto &entry : std::filesystem::recursive\_directory\_iterator(directory))

{

if (entry.is\_regular\_file() && entry.path().filename() == filename)

{

std::cout << "找到文件: " << entry.path() << std::endl;

}

}

}

// 文件复制

bool copyFile(const std::string &source, const std::string &destination)

{

if (std::filesystem::exists(source))

{

try

{

std::filesystem::copy(source, destination);

return true;

}

catch (std::filesystem::filesystem\_error &e)

{

std::cerr << "复制文件时出错：" << e.what() << std::endl;

}

}

else

{

std::cerr << "源文件不存在: " << source << std::endl;

}

return false;

}

// 文件重命名

bool renameFile(const std::string &oldName, const std::string &newName)

{

if (std::filesystem::exists(oldName))

{

try

{

std::filesystem::rename(oldName, newName);

return true;

}

catch (std::filesystem::filesystem\_error &e)

{

std::cerr << "重命名文件时出错：" << e.what() << std::endl;

}

}

else

{

std::cerr << "原文件不存在: " << oldName << std::endl;

}

return false;

}

};

int main()

{

FileSystem fs;

std::string filename, directory, inputFile, outputFile, oldName, newName,file;

std::vector<std::string> files;

while (true)

{

std::cout << "请选择要执行的操作(输入对应数字):" << std::endl;

std::cout << "1. 创建文件" << std::endl;

std::cout << "2. 显示文件内容" << std::endl;

std::cout << "3. 列出目录内容" << std::endl;

std::cout << "4. 删除文件" << std::endl;

std::cout << "5. 加密文件" << std::endl;

std::cout << "6. 解密文件" << std::endl;

std::cout << "7. 合并文件" << std::endl;

std::cout << "8. 搜索文件" << std::endl;

std::cout << "9. 复制文件" << std::endl;

std::cout << "10. 重命名文件" << std::endl;

std::cout << "11. 退出" << std::endl;

int choice;

std::cin >> choice;

switch (choice)

{

case 1:

std::cout << "请输入要创建的文件名: ";

std::cin >> filename;

if (fs.createFile(filename))

{

std::cout << "文件创建成功!" << std::endl;

}

else

{

std::cout << "文件创建失败!" << std::endl;

}

break;

case 2:

std::cout << "请输入要显示的文件名: ";

std::cin >> filename;

fs.displayFile(filename);

break;

case 3:

std::cout << "请输入要列出的目录路径: ";

std::cin >> directory;

fs.listDirectory(directory);

break;

case 4:

std::cout << "请输入要删除的文件名: ";

std::cin >> filename;

if (fs.deleteFile(filename))

{

std::cout << "文件删除成功!" << std::endl;

}

else

{

std::cout << "文件删除失败!" << std::endl;

}

break;

case 5:

std::cout << "请输入要加密的文件路径: ";

std::cin >> inputFile;

std::cout << "请输入加密后的输出文件路径: ";

std::cin >> outputFile;

if (fs.encryptFile(inputFile, outputFile))

{

std::cout << "文件加密成功!" << std::endl;

}

else

{

std::cout << "文件加密失败!" << std::endl;

}

break;

case 6:

std::cout << "请输入要解密的文件路径: ";

std::cin >> inputFile;

std::cout << "请输入解密后的输出文件路径: ";

std::cin >> outputFile;

if (fs.decryptFile(inputFile, outputFile))

{

std::cout << "文件解密成功!" << std::endl;

}

else

{

std::cout << "文件解密失败!" << std::endl;

}

break;

case 7:

std::cout << "请输入要合并的文件路径(用空格分隔): ";

while (std::cin >> file)

{

files.push\_back(file);

if (std::cin.peek() == '\n')

break;

}

std::cout << "请输入合并后的输出文件路径: ";

std::cin >> outputFile;

if (fs.mergeFiles(files, outputFile))

{

std::cout << "文件合并成功!" << std::endl;

}

else

{

std::cout << "文件合并失败!" << std::endl;

}

files.clear();

break;

case 8:

std::cout << "请输入要搜索的目录路径: ";

std::cin >> directory;

std::cout << "请输入要搜索的文件名: ";

std::cin >> filename;

fs.searchFile(directory, filename);

break; // Add a break statement here

case 9:

std::cout << "请输入要复制的源文件路径: ";

std::cin >> inputFile;

std::cout << "请输入要复制到的目标文件路径: ";

std::cin >> outputFile;

if (fs.copyFile(inputFile, outputFile))

{

std::cout << "文件复制成功!" << std::endl;

}

else

{

std::cout << "文件复制失败!" << std::endl;

}

break;

case 10:

std::cout << "请输入要重命名的文件路径: ";

std::cin >> oldName;

std::cout << "请输入新的文件名: ";

std::cin >> newName;

if (fs.renameFile(oldName, newName))

{

std::cout << "文件重命名成功!" << std::endl;

}

else

{

std::cout << "文件重命名失败!" << std::endl;

}

break;

case 11:

std::cout << "谢谢使用,再见!" << std::endl;

return 0;

default:

std::cout << "无效的选择,请重试." << std::endl;

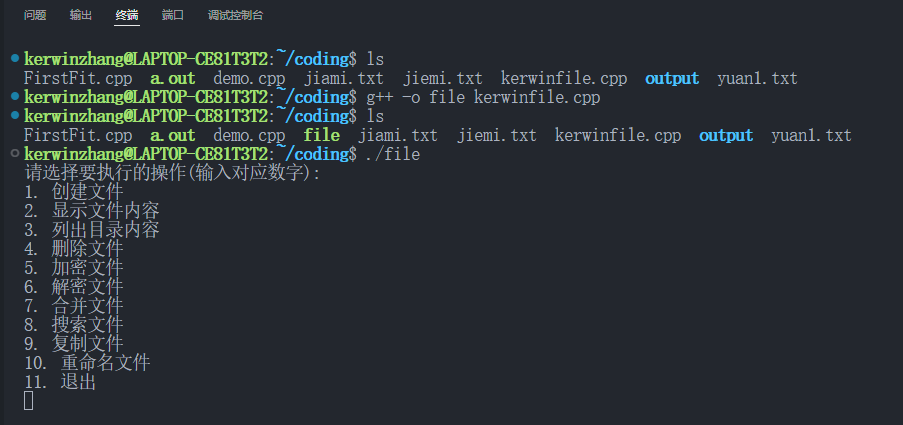
break;

}

}

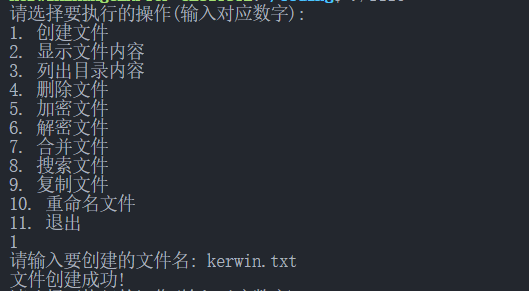
}

**编译过程：**

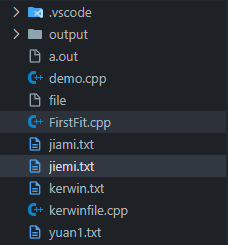


**运行过程以及截图：**

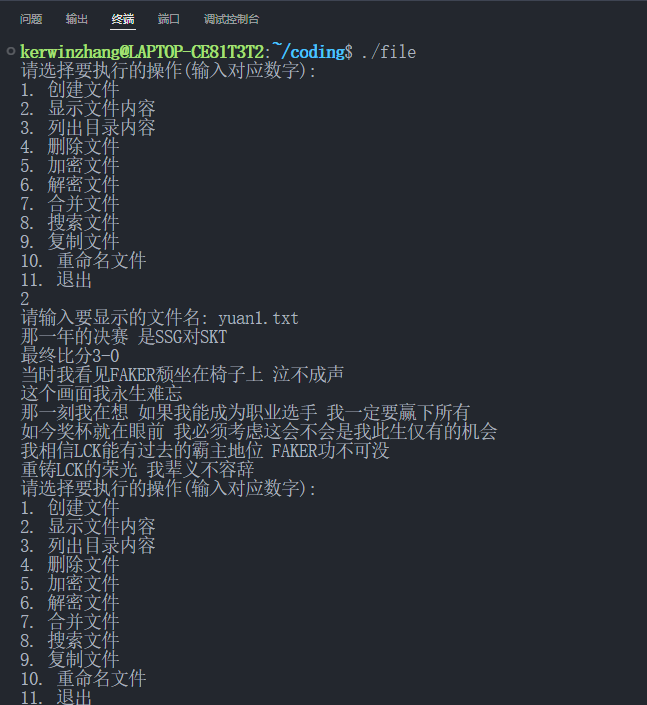
**操作一 创建文件kerwin.txt**



**结果：**



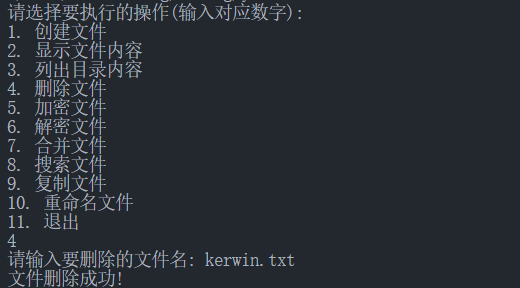
**操作二 显示文件内容：**



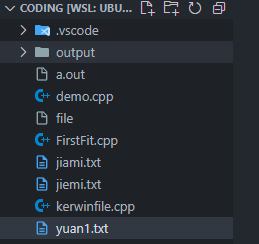
**操作三 列出目录内容：**



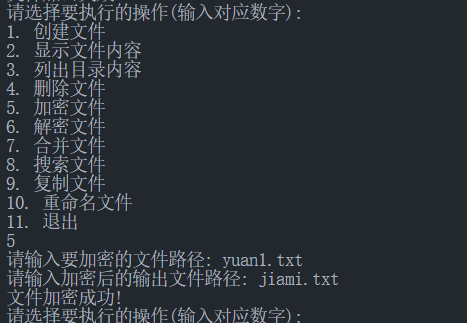
**操作四 删除文件：**



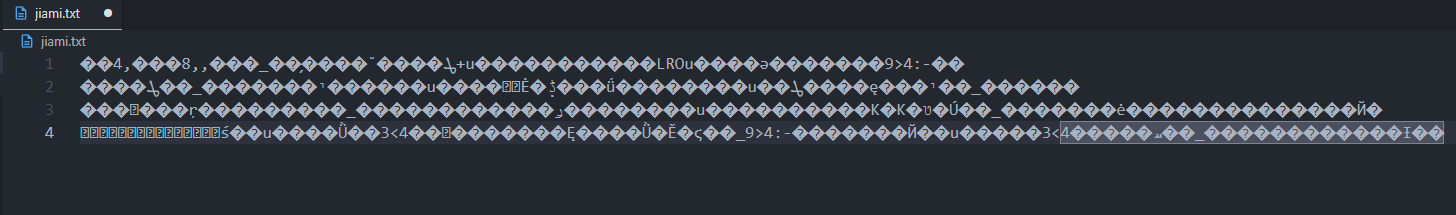
**删除结果;**



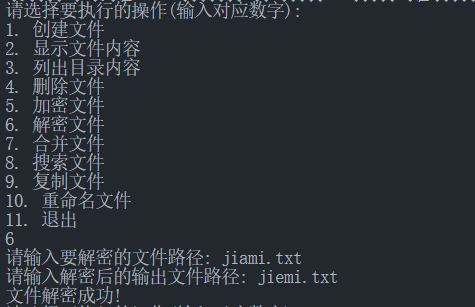
**操作五 加密文件：**



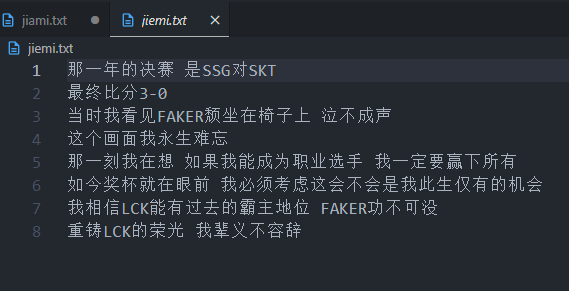
**加密结果;**



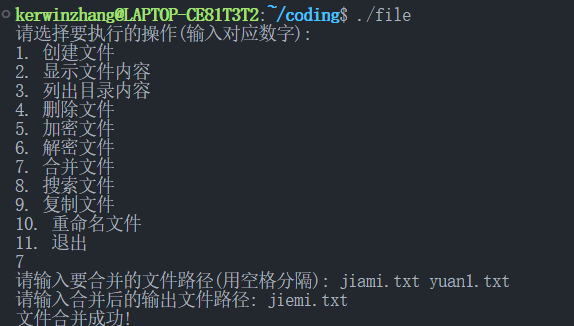
**操作六 解密文件：**



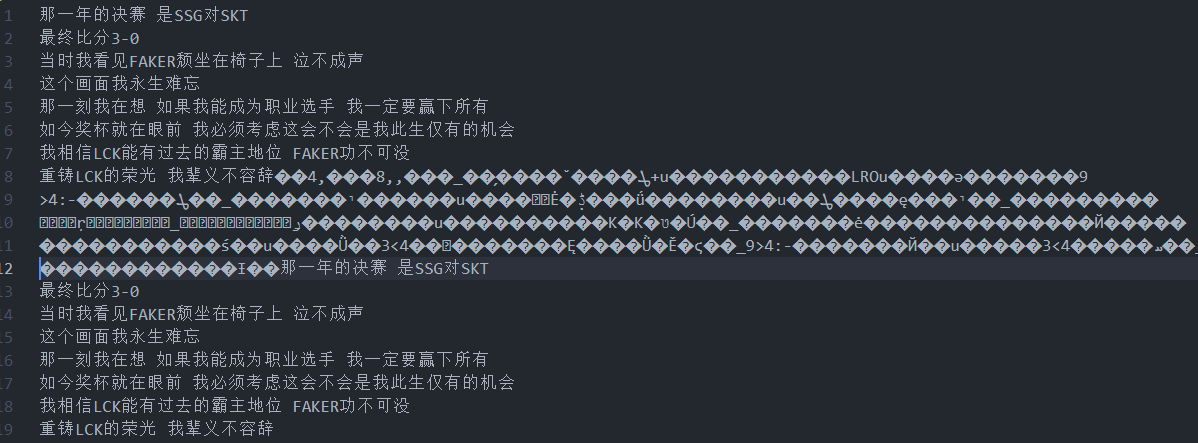
**解密结果：**



**操作七 合并文件：**

****

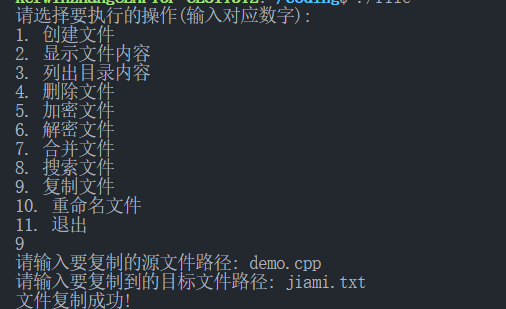
**合并结果：**



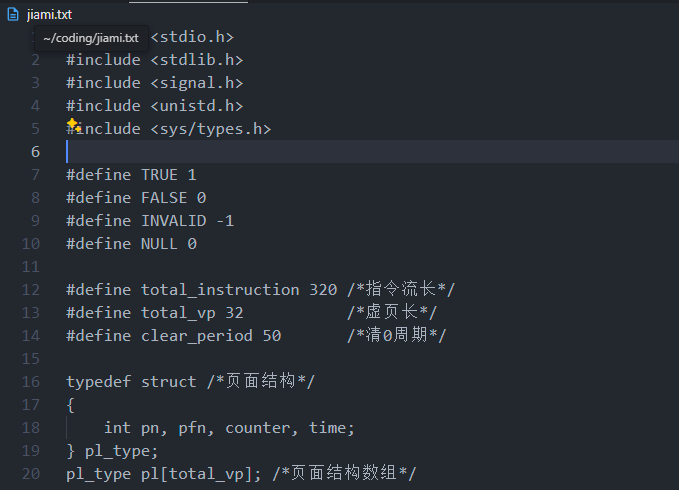
**操作八 搜索文件：**

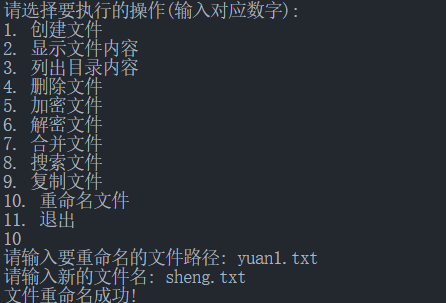


**操作九 复制文件 ：**

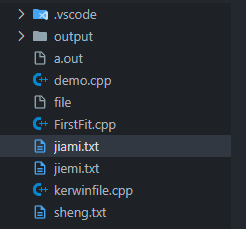


**复制结果：**



**操作十 重命名文件 ：**

**重命名结果：**



5、编程实现一个二级（多用户）文件系统，要求列出程序编译、运行过程以及执行结果的部分截屏。（附加题）注：选做该题的同学，不用做前面的题，直接提交这一道题的程序和结果即可。

三、实验总结和体会

实验总结

本次实验围绕“构建与理解一级文件系统及其基本操作”展开，旨在通过理论学习与实际编码实践，深入探索文件系统的核心功能及其在Linux环境下的具体实现。经过一系列精心设计的实验环节，不仅巩固了对文件系统基本原理的认识，还亲手模拟实现了一个简化版的一级文件系统，涵盖了文件的生命周期管理，包括创建、加密/解密、显示、列目录、删除、合并、查询、复制、重命名等关键操作。

在理论学习部分，通过对文件系统架构的深入解析，我认识到文件系统作为连接用户与物理存储的桥梁，其高效、可靠的数据管理机制对于现代计算至关重要。Linux系统调用的学习则让我掌握了与文件操作紧密相关的API，这些底层接口的灵活运用为后续的模拟实现奠定了坚实的基础。

在实践环节，从设计数据结构到编码实现各功能模块，每一步都充满了挑战与收获。通过定义inode结构体来模拟文件元数据管理，以及构建文件分配表来跟踪文件块的分配状态，我深刻理解了文件系统如何在幕后组织和访问文件。特别是在实现文件加密与解密功能时，虽采用了较为简单的加密算法，却也让我对数据保护的重要性有了直观感受。此外，处理文件合并、查询等操作时遇到的边界条件和异常情况，锻炼了我的问题解决能力和代码调试技巧。

个人体会

这次实验不仅是对操作系统课程知识的一次综合应用，更是一次宝贵的工程实践经历。它让我意识到理论知识与实际应用之间的距离，以及将抽象概念转化为可运行代码所必需的细致思考与反复调试。通过亲自编码实现一个简化版的文件系统，我体验到了软件开发的全过程，从需求分析到系统设计，再到编码实现和测试验证，每一个环节都不可或缺。

尤为值得一提的是，实验过程中遇到的各种问题，如内存管理、错误处理策略的选择等，促使我查阅更多资料，不断优化设计方案，这种解决问题的过程极大地提升了我的自学能力和独立思考能力。同时，团队合作讨论部分功能的实现方案，也让我学会了如何有效沟通技术想法，以及如何在团队协作中发挥自己的优势。

总之，此次实验不仅增强了我对操作系统底层机制的理解，更重要的是，它培养了我的实践动手能力和问题解决策略，为将来从事相关领域的深入研究和技术开发打下了坚实的基础。未来，我期待能够继续探索操作系统更深层次的奥秘，将所学知识应用于更复杂的系统设计与实现中。