

# 操作系统课设报告

设计日期：2025 年 6 月 10 日

# 目录

目录 .....	2
1 引言 .....	3
2 需求分析.....	4
2.1 用户管理模块.....	4
2.1.1 用户注册.....	4
2.1.2 用户登录.....	4
2.1.3 登录会话管理.....	4
2.2 文件管理模块.....	4
2.2.1 文件类操作.....	4
2.2.2 目录类操作.....	4
2.3 本地文件交互功能.....	5
2.4 多线程并发支持.....	5
2.5 目录树功能.....	5
2.6 系统持久性与虚拟磁盘格式.....	5
3. 系统设计.....	5
3.1 总体设计.....	5
3.2.1 协议设计.....	6
3.2.2 Socket 通信流程.....	6
3.3 客户端模块划分.....	6
3.3.1 网络通信模块.....	6
3.3.2 用户管理模块.....	6
3.3.3 用户界面处理模块.....	7
3.4 服务端模块.....	7
3.4.1 网络通信模块.....	7
3.4.2 用户状态与登录模块.....	7
3.4.3 虚拟磁盘与 FAT 管理模块.....	7
3.4.4 文件与目录操作模块.....	7
3.4.5 导入导出模块.....	8
3.4.6 多线程并发模块.....	8
4. 系统结果演示.....	8
4.1 用户登录注册.....	8
4.2 文件进行操作.....	9
4.3 对文件的导入导出.....	10
4.4 树状图展示.....	10
5. 总结.....	10

# 1 引言

在信息化社会高速发展的背景下，操作系统作为计算机系统的核心软件基础设施，其架构设计与实现机制始终是计算机科学与技术领域的的前沿研究方向。文件系统作为操作系统资源管理的核心组件，不仅承担着持久化存储设备与上层应用之间的数据交互中介角色，更通过抽象化的文件组织方式为用户提供了逻辑化的数据管理视图。随着云计算、分布式存储等技术的兴起，现代文件系统正面临着多用户并发访问、异构设备兼容、细粒度权限控制等全新挑战，这对传统单用户文件系统的设计范式提出了革命性要求。

本次课程设计聚焦于构建一个虚拟文件系统教学原型，旨在通过系统化的工程实践深化对操作系统内核机制的理解。该系统采用分层架构设计思想，自底向上构建包含物理存储模拟层、虚拟文件管理层、用户接口层的完整技术栈，重点突破多用户隔离与并发控制两大技术瓶颈。在用户管理方面，系统基于两级目录结构实现用户文件系统的逻辑隔离，通过用户身份认证模块与文件系统命名空间的动态绑定，确保不同用户进程拥有独立的文件视图，同时引入基于角色的访问控制模型，支持文件级读写权限的精细化配置。针对并发处理需求，系统采用多线程并发模型提升吞吐量，设计无锁化数据结构优化高并发场景性能。

在功能实现层面，系统完整包括目录树形结构的可视化呈现、文件指针的随机访问控制、缓冲式读写机制、硬链接与符号链接的语义区分以及文件元数据动态管理等核心功能。特别地，系统开发了本地文件系统适配器，实现虚拟文件系统与宿主操作系统文件系统的双向数据映射，支持目录结构的导入/导出操作。

通过内存管理、进程同步、设备抽象等操作系统核心模块的协同实现，培养了从零构建复杂软件系统的工程能力；在多线程环境下验证临界区保护、死锁预防、性能调优等关键技术，深化学生对并发模型的理解；从物理存储介质到逻辑文件视图的层次化抽象过程中，强化学生对计算机系统分层设计哲学的认知。最终交付成果将包含可执行的系统原型、详尽的测试用例集、性能分析报告，以及反映设计决策过程的技术文档，为学生从事操作系统内核开发、分布式存储系统设计等前沿领域工作奠定坚实基础。

## 2 需求分析

### 2.1 用户管理模块

系统需构建完整的用户生命周期管理体系，所有文件操作均绑定至独立用户上下文，确保多用户间数据完全隔离。

#### 2.1.1 用户注册

系统应允许用户注册新账号。注册时通过遍历本地用户列表文件，实时校验用户名唯一性。若检测到重复，需返回明确错误提示

注册信息（用户名和密码）应持久化保存，具体保存为 `userfile.txt`。

#### 2.1.2 用户登录

用户通过输入用户名与密码登录系统。

每次登录必须验证用户身份信息，读取本地的 `userfile.txt` 文件和用户名和密码进行匹配，如果输入的用户名不存在，报错用户名不存在，如果输入的密码错误则会相应的报错并开始计数。

若密码连续输错超过 3 次，该用户账户临时锁定。

#### 2.1.3 登录会话管理

登录进行当前用户文件的管理，输入命令并把命令传给服务端进程。

登录状态在当前程序运行周期内持续有效，用户退出登录或关闭程序视为会话结束。

### 2.2 文件管理模块

系统支持以下 16 个命令。

#### 2.2.1 文件类操作

<code>create</code>	创建新文件，写入空内容或默认内容
<code>delete</code>	删除指定文件，释放空间
<code>open / close</code>	打开或关闭文件句柄
<code>read / write</code>	从打开的文件读取内容或写入数据
<code>move / copy</code>	移动或复制文件至其他目录或重命名
<code>lock</code>	加锁文件，使其他进程不可访问
<code>head -num</code>	显示文件前 <code>num</code> 行
<code>tail -num</code>	显示文件末尾 <code>num</code> 行
<code>lseek</code>	移动读写指针以支持随机访问

#### 2.2.2 目录类操作

<code>cd</code>	进入某个目录（支持相对/绝对路径）
<code>dir</code>	显示当前目录下文件与子目录
<code>mkdir</code>	创建子目录

rmdir          非递归的删除空目录

## 2.3 本地文件交互功能

import <本地路径> <目标目录>: 从本地磁盘导入文件至虚拟磁盘

export <文件名> <本地路径>: 将虚拟磁盘文件导出至本地磁盘

## 2.4 多线程并发支持

系统至少包含两个线程:

前端线程: 与用户交互, 接收并解释命令

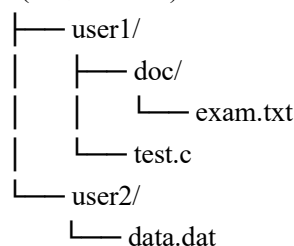
后端线程: 维护虚拟磁盘数据的一致性与访问

支持多个客户端(进程)并发访问共享磁盘, 每个客户端可操作, 磁盘维护由唯一线程统一管理

## 2.5 目录树功能

显示当前用户目录下的完整结构树, 展示文件/文件夹的大小。支持嵌套层级展示

如: / (虚拟根目录)



## 2.6 系统持久性与虚拟磁盘格式

使用二进制文件保存虚拟磁盘内容, 系统关闭后再次运行仍能恢复上次的文件系统状态。

# 3. 系统设计

## 3.1 总体设计

对系统总体采取客户端-服务器的模式, 系统以命令行为交互界面, 用户输入命令后由客户端解析并通过 socket 端口号传递至本地服务端, 由服务端统一操作虚拟磁盘数据, 完成系统功能。



## 3.2 信息传递模型

### 3.2.1 协议设计

协议设计如下：

请求格式：<命令类型>-<参数 1>-<参数 2>

响应格式：<返回消息或数据内容>

客户端与服务器之间采用自定义的基于 TCP 的命令协议进行通信，请求和响应的格式通过字符串拼接和分割实现。客户端发送命令时，将命令类型和参数用 - 分隔；服务器接收后，解析命令并执行相应操作，然后将结果以字符串形式返回。

### 3.2.2 Socket 通信流程

- 客户端使用 TCP Socket（127.0.0.1:8080）连接服务器；
- 登录后保持会话；
- 每个命令通过 Socket 发送至服务器；
- 服务器处理后返回结果。

## 3.3 客户端模块划分

客户端主要包含以下四个模块。

### 3.3.1 网络通信模块

创建套接字：创建一个 TCP 客户端，并尝试连接到本地的服务器。通过 socket 函数创建一个基于 IPv4 和 TCP 的套接字。如果套接字创建失败，就输出错误信息并清理资源。程序设置服务器的地址信息，将地址族设为 AF\_INET，端口号用 htons 转换成网络字节序，然后使用 inet\_pton 将字符串形式的 IP 地址转换为网络格式，填充到 serverAddress 结构中。然后调用 connect 尝试与指定地址和端口的服务器建立连接。如果连接失败，同样会输出错误码并进行资源清理。

从服务器端接受消息：使用 recv()函数接收服务器发送的消息，将接收到的消息存储到全局缓冲区 buffer 中。

从客户端发送消息：使用 send()函数将消息字符串发送到服务器。检查发送是否成功，若失败则输出错误信息。

给服务端发送的 ack：调用上述的 sendtoserver()函数将 ack 发给服务端，供服务端进行下一步操作。

关闭套接字并清理资源：调用 closesocket()关闭套接字，调用 WSACleanup()清理 Winsock 资源。

### 3.3.2 用户管理模块

用户注册：打开存储用户信息的文件 userfile.txt 尝试注册，直到用户成功注册或主动选择退出。在注册过程中，首先输入一个用户名。为了防止用户名重复，它会打开用户信息文件，逐行读取每一个已有的用户名与密码，并通过提取用户名部分，与当前输入的用户名进行比对。如果发现用户名已经存在，程序会提示用户该用户名已被占用。如果输入的用户名未被使用。需连续输入两次密码以确保一致性。如果两次密码完全一致，程序会将该用户名与密码以空格分隔的形式写入用户信息文件中，并提示注册成功。相反，如果两次输入不一致，系统会提示错误并让用户选择是否重新尝试。

用户登录：让用户可以多次尝试登录。用户输入用户名，之后程序会打开用户信息文件并逐行读取每一条记录，如果用户名存在，系统会进入密码验证流程。用户最多有三次输入密码的机会。每次输入密码后，记录的密码进行比对。如果匹配成功，说明身份验证通过，此时系统调用 `menu(clientSocket)` 函数进入主菜单功能模块，表示登录成功；如果密码错误，则提示用户是否重新输入，如果连续三次错误，系统会终止登录流程。如果用户名不存在，系统会提示“未找到该用户”。

### 3.3.3 用户界面处理模块

对命名的处理和发送，以 `create` 为例子：对用户命令的读取，并对命令进行加工调用 `sendtoserver()` 函数发给服务端，并接收服务端的发回，后对服务端发送确定 `ack` 并打印服务端发来的信息。

## 3.4 服务端模块

服务端作为虚拟文件系统的控制核心，负责管理用户登录、命令解析、磁盘访问与文件操作等全部系统功能。整体设计采用模块化结构，结合多线程机制，支持多用户并发访问，每位用户拥有独立的虚拟磁盘资源和命名空间。服务端模块主要划分为以下六个子模块。

### 3.4.1 网络通信模块

本模块负责搭建服务器与客户端之间的通信桥梁，实现 `TCP Socket` 连接的创建、监听与会话管理。服务端启动后，会创建一个基于 `TCP` 的服务器套接字，并绑定至指定本地地址（如 `127.0.0.1`）和端口号。随后进入监听状态，持续等待来自客户端的连接请求。每当有客户端连接时，服务端即为该连接分配一个独立线程，专门负责该客户端的命令接收、解析与反馈。在每个线程内部，服务端会持续等待客户端发送的命令字符串，并根据命令类型进行解析、路由和处理。服务端通过简单的消息协议接收请求内容并返回结果，通信过程中采用字符串作为消息载体，传输高效，结构清晰。

### 3.4.2 用户状态与登录模块

本模块用于管理每个客户端用户的登录状态和身份绑定，确保不同用户操作互不干扰，实现命名空间隔离。当客户端发起登录请求后，服务端会从用户信息文件中查找对应账号，并读取其个人虚拟磁盘数据。在登录成功后，服务端会将当前线程上下文与该用户绑定，包括用户名、用户磁盘名、当前路径、当前目录等信息。模块确保每个客户端线程处理时只访问当前用户所属的虚拟磁盘区域，其他用户数据对其不可见，从而实现用户数据的逻辑隔离与安全访问。

### 3.4.3 虚拟磁盘与 FAT 管理模块

本模块负责虚拟磁盘的初始化、读取、更新与保存操作，是系统数据存储的核心。每个用户在第一次登录时，会自动分配一个独立的虚拟磁盘文件，用于保存其文件数据与结构信息。虚拟磁盘以文件形式保存在本地磁盘上，内部结构遵循 `FAT`（文件分配表）方式组织。在系统运行期间，服务端通过内存映射方式将虚拟磁盘载入到内存缓冲区，通过 `FAT` 表维护磁盘块的使用情况。所有文件与目录的实际数据都存储在磁盘数据区，通过 `FAT` 表实现块链式结构的连接。系统支持多次挂载与保存，确保虚拟磁盘状态在程序重启后依然一致。

### 3.4.4 文件与目录操作模块

该模块实现了文件系统的全部指令集，包括文件的创建、打开、读取、写入、复制、移动、删除等基本操作，以及目录的进入、创建、删除与遍历等功能。文件操作基于模拟的文

件控制块（FCB）结构进行管理，每个文件或目录对应一个控制块，记录其名称、大小、起始块号、类型和修改时间等信息。目录则以树形结构组织，支持嵌套与递归遍历。服务端收到客户端指令后，会调用本模块相应函数进行处理，并返回操作结果或数据内容。文件指针控制、加锁机制、目录路径解析、路径合法性判断等也由本模块负责。

### 3.4.5 导入导出模块

导入导出模块实现本地文件系统与虚拟磁盘之间的数据交换。客户端可通过 `import` 命令将本地文件写入服务器端虚拟磁盘，也可通过 `export` 命令将虚拟磁盘内的文件导出为本地文件。在导入操作中，系统会读取本地文件内容，自动分配虚拟磁盘空间并写入，同时更新 FAT 表与控制块。在导出操作中，服务端会定位指定文件，读取数据后写入目标路径，实现虚拟文件向本地系统的映射输出。该模块扩展了虚拟文件系统的应用边界，为测试与验证提供了便利条件。

### 3.4.6 多线程并发模块

为支持多客户端同时访问，系统设计了基于线程的并发模型。每当有新客户端连接，服务端便创建一个独立线程，与该客户端建立私有通信通道并维护其操作上下文。每个线程持有各自的用户状态信息（用户名、当前路径、虚拟磁盘数据等），执行命令时仅作用于自身环境，线程间彼此独立，互不干扰。此外，系统在文件操作中引入加锁机制，用于防止并发访问同一文件时发生数据冲突。在读写过程中，服务端可判断文件是否处于锁定状态，决定是否允许当前线程操作，从而提高系统的一致性与稳定性。

## 4. 系统结果演示

### 4.1 用户登录注册

```
register
请输入注册用户名：lu
用户名已存在！
是否重新输入（y/n）？
y
请输入注册用户名：jia
请输入密码：222
请再次输入密码：222
注册成功！
|
```

对于已存在的用户不让注册，注册输入两遍密码确认。



```

请登录或注册
login
用户名:jia
密码:3
输入密码错误!
是否重新输入 (y/n)?:y
密码:3
输入密码错误!
是否重新输入 (y/n)?:y
密码:3
输入密码错误已达到3次!

```

对于登录输入密码错误达到三次锁定该用户。

## 4.2 文件进行操作

```

jia:\>mkdir main
目录创建成功!
jia:\>cd main
jia:\main\>create a.txt
文件创建成功!
jia:\main\>open a.txt
a.txt打开成功
jia:\main\>write a.txt
请输入文件内容, 并以 '#' 为结束标志
hello
world#
写入完成!
jia:\main\>read a.txt
-----文件内容: -----
hello
world
jia:\main\>tail a.txt 1
行数请以-num类型输入!
jia:\main\>tail a.txt -1
-----文件内容: -----
world
jia:\main\>head a.txt -1
-----文件内容: -----
hello
jia:\main\>copy a.txt new.txt
文件已打开, 请先关闭
jia:\main\>close a.txt
a.txt关闭成功
jia:\main\>copy a.txt new.txt
复制成功
jia:\main\>dir

```

文件名	类型	起始磁盘块号	大小	打开状态	加锁状态	lseek指针位置
a.txt	<FILE>	11	11	未打开	未加锁	0
new.txt	<FILE>	12	11	未打开	未加锁	0

```

jia:\main\>cd ..
jia:\>

```

```

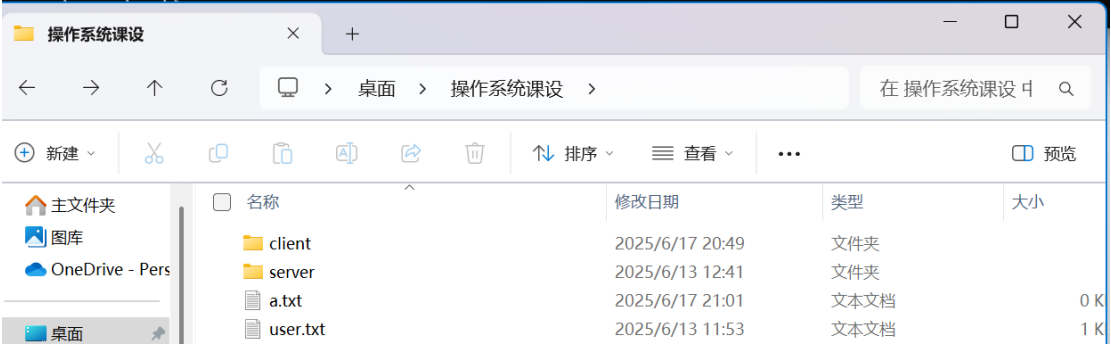
jia:\main\>create a.txt
文件创建成功!
jia:\main\>lock a.txt
a.txt加锁成功
jia:\main\>open a.txt
文件已加锁, 请先解锁
jia:\main\>

```

```
jia:\main\>move a.txt er\
移动成功
jia:\main\>cd er
jia:\main\er\>dir
文件名                类型                起始磁盘块号      大小      打开状态  加锁状态  lseek指针位置
a.txt                  <FILE>             17                0          未打开    未加锁    0
```

## 4.3 对文件的导入导出

```
jia:\main\er\>export a.txt C:\Users\卢家璐\Desktop\操作系统课设\a.txt
导出文件成功!
```



```
jia:\main\>import C:\Users\卢家璐\Desktop\操作系统课设\user.txt main
导入文件成功!
jia:\main\>dir
文件名                类型                起始磁盘块号      大小      打开状态  加锁状态  lseek指针位置
user.txt              <FILE>             16                7          未打开    未加锁    0
jia:\main\>
```

## 4.4 树状图展示

```
jia:\>tree
Tree structure of directory:
├─ main/ (DIR)
├─ a.txt (FILE, Size: 0 bytes)
└─ user.txt (FILE, Size: 7 bytes)
```

# 5. 总结

通过本次操作系统课程设计，我对文件系统的内部机制和操作系统内核功能有了更加深入的理解。整个项目围绕“多用户虚拟文件系统”的主题，从用户管理、命令解析、磁盘模拟、到多线程并发处理，完整地实现了一个具有基础功能的教学系统。在动手实现的过程中，我逐步掌握了 TCP Socket 通信、FAT 表管理、用户登录会话控制、文件读写与权限控制等关键技术点，也体会到了操作系统中模块划分、数据结构设计和系统稳定性优化的重要性。

在客户端与服务端的搭建过程中，我遇到了诸如并发访问冲突、命令解析错误、文件路径管理混乱等问题，但通过不断调试和查阅资料，逐步解决了这些挑战。这让我深刻认识到理论知识和实际编码之间的差距，同时也提升了我分析问题和解决问题的能力。

本系统最终实现了多用户隔离、文件与目录的基本操作、命令行交互、目录树结构展示以及虚拟磁盘的持久化保存等核心功能，具备了一定的实用性与可扩展性。我认为这次课设不仅巩固了对操作系统核心知识的掌握，也锻炼了我从零开始设计和实现一个小型系统的工程实践能力。

这次课程设计是一次非常有意义的学习和锻炼机会，不仅提升了我的综合编程能力，也为今后学习更复杂的系统原理和分布式存储技术打下了基础。