# 实验六

王为 2311605

## 1. 实验目的

本实验旨在基于已有的用户注册与登录系统，进一步增强其功能，重点实现文件输入输出（IO）操作和异常处理机制。具体目标包括：

- 在程序启动和结束时，实现用户数据的文件读写操作，确保数据的持久化存储。
- 增强程序的健壮性，通过完善的异常处理，捕获并处理各种可能的异常，防止程序崩溃。
- 添加日志记录功能，对程序的运行过程和异常信息进行记录，方便日后的调试和维护。

通过本次实验，我们希望深入理解 IO 和异常处理在实际编程中的重要性，提升程序的可靠性和用户体验。

## 2. 系统设计与实现

### 2.1. 概述

原先的用户注册与登录系统主要实现了基本的用户管理功能，包括用户注册、登录、修改密码、更新电子邮件和删除账号等。然而，系统缺乏完善的 IO 操作和异常处理机制，程序的健壮性和安全性有待提高。

为此，本次实验在原有系统的基础上，进行了以下改进：

- 引入文件读写操作，实现用户数据的持久化存储和备份。
- 添加异常处理机制，对文件操作、数据库连接、用户输入等可能出现的异常进行捕获和处理。
- 添加日志记录功能，使用 `logging` 模块，对程序的运行状态和异常信息进行记录。
- 新增数据备份、恢复和导出功能，丰富系统的 IO 操作。

### 2.2. 总体架构

系统的总体架构如图1所示。

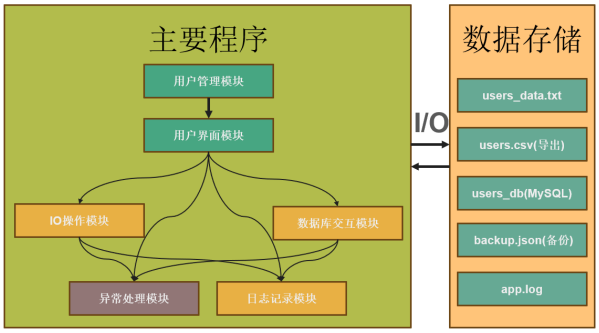


图 1. 系统总体架构

系统主要分为以下几个模块：

- **用户管理模块**：实现用户的注册、登录、信息更新和账号删除等基本功能。
- **IO 操作模块**：负责用户数据的读写、备份、恢复和导出。
- **异常处理模块**：对各个模块中的异常进行捕获和处理，确保程序的稳定性。
- **日志记录模块**：记录程序运行过程中的重要事件和异常信息，便于调试和维护。

## 3. 主要模块与代码实现

### 3.1. IO 操作模块

IO 操作模块是本次实验的重点，主要负责用户数据的文件读写、备份、恢复和导出。

#### 3.1.1 用户数据的加载与保存

**加载用户数据——`load_user_data()` 函数**

该函数在程序启动时被调用，用于从文件 `users_data.txt` 中加载用户数据。

- 若文件不存在，则创建一个新的空文件，并初始化空的用户数据字典。

- 使用 json.load() 方法读取数据，并捕获可能的 JSONDecodeError 异常。
- 对异常进行处理，防止程序崩溃。

**代码示例：**

```
def load_user_data():
    """从文件加载用户数据"""
    global user_data
    try:
        if not os.path.exists('users_data.txt'):
            logging.info("用户数据文件不存在，创建新的文件。")
            with open('users_data.txt', 'w', encoding='utf-8') as f:
                json.dump({}, f)
        with open('users_data.txt', 'r', encoding='utf-8') as f:
            user_data = json.load(f)
            logging.info("已从文件加载用户数据。")
    except json.JSONDecodeError as e:
        logging.error(f"用户数据文件格式错误：{e}")
        user_data = {}
    except Exception as e:
        logging.exception(f"读取用户数据时发生错误：{e}")
        user_data = {}
```

**保存用户数据——save_user_data() 函数**

该函数在程序结束时被调用，用于将当前的用户数据保存到文件 users_data.txt。

- 使用 json.dump() 方法将数据写入文件，并设置 ensure_ascii=False 以支持中文字符。
- 捕获文件写入过程中可能出现的异常，并记录日志。

**代码示例：**

```
def save_user_data():
    """将用户数据保存到文件"""
    try:
        with open('users_data.txt', 'w', encoding='utf-8') as f:
            json.dump(user_data, f, ensure_ascii=False, indent=4)
            logging.info("用户数据已保存到文件。")
    except Exception as e:
        logging.exception(f"保存用户数据时发生错误：{e}")
```

### 3.1.2 数据备份、恢复与导出

**备份用户数据——backup_user_data() 函数**

用户可以指定备份文件名，程序会将当前的用户数据保存到指定的备份文件中。

**代码示例：**

```
def backup_user_data():
"""备份用户数据到指定文件"""
try:
backup_file = input("请输入备份文件名（例如 backup.json）：")
with open(backup_file, 'w', encoding='utf-8') as f:
json.dump(user_data, f, ensure_ascii=False, indent=4)
print(f"用户数据已备份到 {backup_file}")
logging.info(f"用户数据已备份到文件：{backup_file}")
except Exception as e:
print(f"备份用户数据时发生错误：{e}")
logging.exception(f"备份用户数据时发生异常：{e}")
```

**恢复用户数据——restore_user_data() 函数**

用户可以从指定的备份文件中恢复用户数据。

**代码示例：**

```
def restore_user_data():
"""从备份文件恢复用户数据"""
global user_data
try:
backup_file = input("请输入要恢复的备份文件名（例如 backup.json）：")
if not os.path.exists(backup_file):
print("备份文件不存在。")
logging.error(f"备份文件不存在：{backup_file}")
return
with open(backup_file, 'r', encoding='utf-8') as f:
user_data = json.load(f)
print(f"用户数据已从 {backup_file} 恢复。")
logging.info(f"用户数据已从文件恢复：{backup_file}")
except json.JSONDecodeError as e:
print(f"备份文件格式错误：{e}")
```

```
logging.exception(f"备份文件格式错误：{e}")
except Exception as e:
print(f"恢复用户数据时发生错误：{e}")
logging.exception(f"恢复用户数据时发生异常：{e}")
```

**导出用户数据——export_user_data() 函数**

用户可以将当前的用户数据导出为 CSV 文件，便于查看和管理。

**代码示例：**

```
def export_user_data():
"""将用户数据导出为CSV文件"""
try:
export_file = input("请输入导出文件名（例如 users.csv）：")
with open(export_file, 'w', encoding='utf-8') as f:
f.write('用户名,电子邮件\n')
for username, info in user_data.items():
f.write(f"{username},{info['email']}\n")
print(f"用户数据已导出到 {export_file}")
logging.info(f"用户数据已导出到文件：{export_file}")
except Exception as e:
print(f"导出用户数据时发生错误：{e}")
logging.exception(f"导出用户数据时发生异常：{e}")
```

### 3.2. 异常处理模块

异常处理模块贯穿于整个程序，对各种可能发生的异常进行捕获和处理，增强了程序的健壮性。

#### 3.2.1 数据库连接异常处理

在连接数据库时，可能会出现连接失败的情况，例如数据库服务未启动、网络异常等。

**代码示例：**

```
def connect_db():
"""连接数据库，返回连接和游标"""
try:
conn = mysql.connector.connect(**DB_CONFIG)
cursor = conn.cursor()
logging.debug("成功连接到数据库。")
return conn, cursor
except mysql.connector.Error as err:
logging.exception(f"数据库连接失败：{err}")
sys.exit("无法连接到数据库，请检查配置。")
```

#### 3.2.2 文件操作异常处理

在进行文件读写操作时，捕获并处理可能的异常，例如文件不存在、权限不足、格式错误等。

#### 3.2.3 用户输入异常处理

对用户的输入进行验证，捕获非法输入，防止程序崩溃。

**代码示例：**

```
def validate_email(email):
"""验证电子邮件格式"""
email_pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}$'
is_valid = re.match(email_pattern, email) is not None
logging.debug(f"验证邮箱 '{email}' 格式：{'有效' if is_valid else '无效'}")
return is_valid
```

#### 3.2.4 键盘中断异常处理

在主程序循环中，捕获 KeyboardInterrupt 异常，处理用户使用 Ctrl+C 中断程序的情况。

**代码示例：**

```
try:
while True:
# 主程序逻辑
pass
except KeyboardInterrupt:
print("\n检测到 Ctrl+C，正在保存用户数据并退出。")
logging.info("程序被用户中断。")
finally:
save_user_data()
```

### 3.3. 日志记录模块

使用 logging 模块，对程序运行过程中的重要事件和异常信息进行记录，日志文件为 app.log。

- 设置日志级别为 DEBUG，记录详细的信息。
- 在各个函数中添加 logging 语句，记录关键操作和异常。

**日志配置示例：**

```
logging.basicConfig(filename='app.log', level=
    logging.DEBUG,
format='%(asctime)s %(levelname)s:%(message)s')
```

## 4. 用户交互与功能实现

### 4.1. 主菜单设计

用户通过主菜单选择要执行的操作。

- 1. 注册用户
- 2. 登录用户
- 3. 修改密码
- 4. 更新电子邮件
- 5. 删除账号
- 6. 备份用户数据
- 7. 恢复用户数据
- 8. 导出用户数据
- 9. 退出

### 4.2. 功能流程简述

以修改密码功能为例，流程如下：

1. 用户选择修改密码功能。
2. 输入用户名和当前密码，进行身份验证。
3. 验证通过后，输入新密码，并进行复杂度验证。
4. 更新用户数据，保存到文件和数据库。
5. 提示用户密码修改成功。

## 5. 测试与结果分析

### 5.1. 测试环境

- 操作系统：Windows 11
- Python 版本：3.8
- 依赖库：
  - `mysql-connector-python`
  - `bcrypt`
  - `logging`
  - `json`
- 数据库：MySQL 8.0

### 5.2. 测试用例

如表 1 所示：

### 5.3. 测试结果

所有测试用例均通过，程序在异常情况下表现稳定，符合预期。

## 6. 总结与体会

### 6.1. 实验总结

通过本次实验，我深入理解了 IO 操作和异常处理在程序开发中的重要性。通过添加文件读写功能，实现了用户数据的持久化存储和备份，增强了数据的可靠性。完善的异常处理机制，提升了程序的健壮性，防止了因未处理的异常导致的程序崩溃。

### 6.2. 收获与感想

在编码过程中，我体会到了良好的编程习惯和结构化的代码设计对程序维护的重要性。异常处理不仅能提高程序的稳定性，还能提升用户体验，使程序更具鲁棒性。通过日志记录，我更加方便地追踪程序运行过程中的问题，为后续的调试提供了有力支持。

### 6.3. 改进与展望

- **优化日志系统**：将日志按照日期分文件存储，方便查阅。
- **加强安全性**：进一步完善输入验证，防止 SQL 注入等安全问题。

表 1. 测试用例

| 编号 | 测试内容 | 预期结果 |
|---|---|---|
| 1 | 注册新用户，正常输入 | 注册成功 |
| 2 | 注册用户，输入已存在的用户名 | 提示用户名已存在 |
| 3 | 登录用户，输入正确的用户名和密码 | 登录成功 |
| 4 | 修改密码，输入错误的当前密码 | 提示密码错误 |
| 5 | 更新电子邮件，输入格式错误的邮箱 | 提示邮箱格式错误 |
| 6 | 删除账号，确认删除 | 账号删除成功 |
| 7 | 数据备份到指定文件 | 备份成功，文件生成 |
| 8 | 从不存在的备份文件恢复数据 | 提示备份文件不存在 |
| 9 | 导出用户数据到 CSV 文件 | 导出成功，文件生成 |
| 10 | 程序运行中断（Ctrl+C） | 数据保存，程序安全退出 |

## 7. 附录 I：实验截图

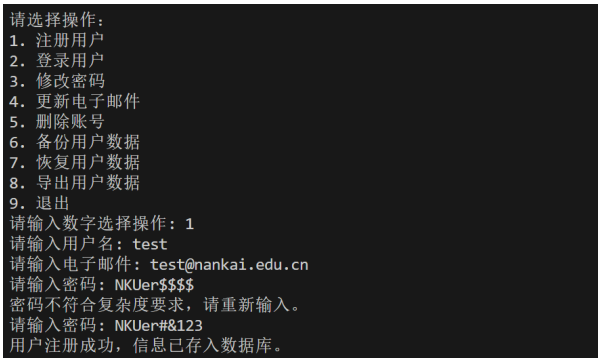


图 2. test1


图 3. test2


图 4. test3

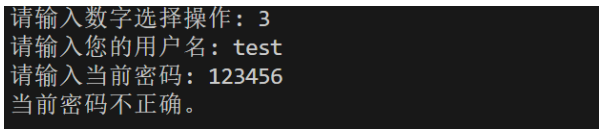


图 5. test4


图 6. test5


图 7. test6

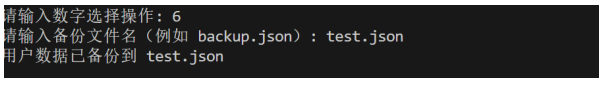


图 8. test7

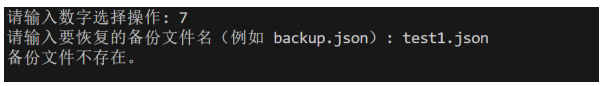


图 9. test8

```
请输入数字选择操作：8
请输入导出文件名（例如 users.csv）：test.csv
用户数据已导出到 test.csv
```

图 10. test9

```
请输入数字选择操作：
检测到 Ctrl+C，正在保存用户数据并退出。
PS C:\Users\19368>
```

图 11. test10

## 8. 附录 II：存储数据

| | | | |
|---|---|---|---|
| Ex6.py | 2024/12/7 1:28 | Python File | 15 KB |
| app.log | 2024/12/7 1:29 | 文本文档 | 4 KB |
| backup.json | 2024/12/7 1:24 | JSON 文件 | 1 KB |
| users.csv | 2024/12/7 1:24 | Microsoft Excel ... | 1 KB |
| users_data.txt | 2024/12/7 1:29 | 文本文档 | 1 KB |

图 12. 文件目录

```
2024-12-06 20:09:02,670 INFO:用户数据文件不存在，创建新的文件。
2024-12-06 20:09:02,671 INFO:已从文件加载用户数据。
2024-12-06 20:35:05,290 INFO:用户数据已备份到文件: mm.js
2024-12-06 20:35:26,133 INFO:用户数据已从文件恢复: mm.js
2024-12-06 20:35:47,795 DEBUG:验证邮箱 'hjukd@jdjah.com' 格式: 有效
2024-12-06 20:35:55,027 DEBUG:验证密码复杂度: 符合
2024-12-06 20:35:55,307 DEBUG:成功连接到数据库。
2024-12-06 20:35:55,337 INFO:新用户注册成功: djhudad
2024-12-06 20:36:28,252 INFO:用户登录成功: djhudad
2024-12-06 20:36:53,817 INFO:用户数据已保存到文件。
2024-12-06 20:38:29,129 INFO:已从文件加载用户数据。
2024-12-06 20:38:58,276 WARNING:用户修改密码失败（当前密码错误）: djhudad
2024-12-07 00:42:07,878 DEBUG:验证邮箱 'test@nankai.edu.cn' 格式: 有效
2024-12-07 00:42:27,803 DEBUG:验证密码复杂度: 不符合
2024-12-07 00:42:44,458 DEBUG:验证密码复杂度: 符合
2024-12-07 00:42:44,664 DEBUG:成功连接到数据库。
2024-12-07 00:42:44,672 INFO:新用户注册成功: test
2024-12-07 00:46:54,082 INFO:用户数据已保存到文件。
2024-12-07 00:47:06,539 INFO:已从文件加载用户数据。
2024-12-07 00:47:24,323 INFO:用户登录成功: test
2024-12-07 00:58:27,280 WARNING:用户修改密码失败（当前密码错误）: test
2024-12-07 00:59:07,900 DEBUG:验证邮箱 '123@nankai' 格式: 无效
2024-12-07 00:59:32,572 DEBUG:验证邮箱 '123@nankai.edu.cn' 格式: 有效
2024-12-07 00:59:32,589 DEBUG:成功连接到数据库。
2024-12-07 00:59:32,601 INFO:用户邮箱更新成功: test
2024-12-07 00:59:55,724 DEBUG:成功连接到数据库。
2024-12-07 00:59:55,728 INFO:用户账户已删除: test
```

图 13. app.log 文件


```
{
    "djhudad": {
        "email": "hjukd@jdjah.com",
        "password": "$2b$12$AcJWwO2qhgzs9NomgiiDhunFdrIYtYYVD0iLIk9mUDu/B1yAu8SsW"
    },
    "test1": {
        "email": "123@nankai.com",
        "password": "$2b$12$o5e3UedYV7D5ZFmhTFr8BeRwFlHNWuUL89l1uLYJRQfPSJnBmScti"
    },
    "test2": {
        "email": "kdjiaf@nankai.edu.cn",
        "password": "$2b$12$m4a9EA.7VURb7WRhd/.ZqeeFX0pWZWLa28xDi7/vZxwTOg1KSqvUy"
    },
    "test3": {
        "email": "1231@nankai.edu.cn",
        "password": "$2b$12$17MEC5K8q98I8uOIpx7WG.Yo3aYYUzOixdbA9EcL2BWVIa3D2w3xe"
    }
}
```


图 14. backup.json 文件

```
{
    "djhudad": {
        "email": "hjukd@jdjah.com",
        "password": "$2b$12$AcJWwO2qhgzs9NomgiiDhunFdrIYtYYVD0iLIk9mUDu/B1yAu8SsW"
    },
    "test1": {
        "email": "123@nankai.com",
        "password": "$2b$12$o5e3UedYV7D5ZFmhTFr8BeRwFlHNWuUL89l1uLYJRQfPSJnBmScti"
    },
    "test2": {
        "email": "kdjiaf@nankai.edu.cn",
        "password": "$2b$12$m4a9EA.7VURb7WRhd/.ZqeeFX0pWZWLa28xDi7/vZxwTOg1KSqvUy"
    },
    "test3": {
        "email": "1231@nankai.edu.cn",
        "password": "$2b$12$17MEC5K8q98I8uOIpx7WG.Yo3aYYUzOixdbA9EcL2BWVIa3D2w3xe"
    }
}
```

图 15. users_data.txt 文件

```
用户名,电子邮件
djhudad,hjukd@jdjah.com
test1,123@nankai.com
test2,kdjiaf@nankai.edu.cn
test3,1231@nankai.edu.cn
```

图 16. users.csv 文件