FTP服务器开发文档

1. 设计概要

1.1 项目概述

简易FTP服务器项目是一个基于openEuler操作系统的数据传输工具，旨在为用户提供一个稳定、安全的文件上传和下载环境。该服务器能够监听指定端口，支持文件的上传和下载操作，并能同时处理多个并发请求，确保用户在多样化的网络环境中能够高效地进行数据交换。本项目特别强调用户权限管理，支持匿名及账号登录，并根据不同的用户角色实现精细化的权限控制。

在功能实现方面，简易FTP服务器项目具备以下特点：

1. 网络监听：服务器能够绑定到指定的网络端口，等待客户端的连接请求。
2. 文件操作：用户可通过服务器进行文件的上传和下载，满足基本的数据传输需求。
3. 并发处理：服务器设计考虑了多线程技术，能够同时处理多个用户的请求，提高系统资源利用率。
4. 用户认证与权限管控：服务器内置匿名账户及特定用户账户（user1、user2），实现基于角色的权限管理，确保数据安全。
5. 脚本支持：项目提供run-ftp.sh脚本，实现自动化安装、配置运行环境，以及编译和运行服务器程序。

根据权限配置要求，本项目实现了以下特定功能：

（1）匿名账户：允许匿名用户查看并下载user1上传的文件，但不支持文件上传操作。

（2）user1账户：具备查看、下载自己上传文件的能力，同时可以下载user2上传的文件。

（3）user2账户：仅能查看和下载自己上传的文件，限制了文件访问范围。

简易FTP服务器项目适用于openEuler 24.03 LTS或麒麟信安Kylinsec V3.5.2操作系统，所有系统配置均采用默认设置，确保项目的通用性和易用性。通过本项目，我们旨在为openEuler用户提供一个简易、高效、安全的FTP解决方案，同时促进仓颉语言的普及和应用。

2. 技术方案

2.1 总体架构

简易FTP服务器项目，基于openEuler操作系统，采用仓颉（Cangjie）编程语言进行开发，从而构建一个高效稳定的数据传输平台。以下为该项目的技术方案概述，采用仓颉语言的特点进行设计：

项目核心架构设计围绕以下几个关键组件：

1. **网络通信框架**：利用仓颉语言的网络编程能力，实现服务器与客户端之间的稳定通信。负责监听端口、处理连接请求，并维持稳定的网络会话。
2. **文件操作库**：使用仓颉语言的文件I/O功能，开发文件上传和下载的接口。该库包括文件读写、权限检查、数据流控制等模块，确保文件操作的安全性和高效性。
3. **并发处理机制**：结合仓颉语言的并发编程特性，设计多线程模型，以支持服务器同时处理多个客户端请求，提升系统的并发处理能力。
4. **认证与权限管理系统**：实现用户认证和权限控制。系统包括用户账户管理、权限分配等功能。
5. **自动化部署脚本**：编写仓颉语言的脚本，用于服务器的自动化部署和启动。脚本将涵盖环境配置、依赖管理、编译构建等流程。

具体技术方案如下：

- **网络通信**：
  - 利用仓颉语言的网络库实现基于 TCP 的可靠数据传输。
  - 开发FTP命令解析器，支持FTP协议的标准命令集。
- **文件操作**：
  - 使用仓颉语言的文件操作API，实现文件的上传和下载逻辑。
- **并发处理**：
  - 利用仓颉语言的并发编程特性，为每个客户端连接创建独立的执行线程。
- **认证与权限管理**：
  - 设计并实现用户认证机制，支持匿名登录和固定账户登录。
  - 实现密码验证，保障账户安全。
  - 根据用户角色设定文件访问权限，实现细粒度的权限控制。
- **自动化部署**：
  - 编写仓颉语言脚本，实现服务器的快速部署和一键启动。
- **系统兼容性**：
  - 确保服务器在 openEuler 及其他兼容操作系统上稳定运行。
  - 支持主流硬件架构，如x86_64和aarch64。

通过采用仓颉语言开发，简易FTP服务器项目将展现其独特的编程优势，提供稳定可靠的数据传输服务，满足用户在 openEuler 系统上的文件传输需求，并确保系统的安全性和易用性。

2.2 功能设计

1. **网络通信模块**

- **端口监听**：

- 服务器应能够监听一个或多个预配置的端口。
- 设计时固定端口为FTP通用端口21。

- **连接管理**：
  - 接受来自客户端的连接请求。
  - 维护一个活跃连接列表，用于管理多个并发会话。
  - 限制最大连接数（backlog），防止资源耗尽。

2. **用户认证模块**

- **匿名登录**：
  - 允许匿名用户登录，但限制其操作权限，仅能下载user1的文件。
- **账号登录**：
  - 支持用户名和密码认证。
  - 由于只需要实现简易的FTP服务器，将用户信息固定在创建FTP服务器时。
- **权限控制**：
  - 根据用户角色分配不同的文件操作权限。

3. **文件操作模块**

- **文件上传**：
  - 客户端能够将文件上传到服务器指定目录。
- **文件下载**：
  - 客户端能够从服务器下载文件。
- **文件管理**：
  - 列出服务器上的文件和目录。
  - 后续：支持创建目录、删除文件、重命名文件等操作。

4. **并发处理模块**

- **多线程支持**：
  - 为每个客户端连接创建独立的线程。
- **同步机制**：
  - 利用仓颉的特性，确保操作的原子性，避免并发访问导致的数据损坏。

5. **日志和监控模块**

- **日志记录**：
  - 记录所有用户操作和系统事件。
- **系统监控**：

- 实时反馈服务器状态。

6. **安全性模块**

- **访问控制**：
  - 限制对特定目录的访问。

7. **配置模块**

- **服务器配置**：
  - 设置服务器参数，如监听端口、最大连接数等。

2.3 主要模块

本项目由几个主要模块构成，从而实现功能的完整性和连贯性。

main：程序的入口

主要功能：

1. 进行项目的初始化工作
2. 包括注册用户信息
3. 初始化数据存放目录
4. 注册用户和对应文件的权限
5. 创建一个FTP服务器实例

auth模块：负责用户相关

主要功能：

1. 注册用户，存入用户名和密码从而维护一个用户列表
2. 进行用户的身份认证
3. 添加用户的权限
4. 获取特定用户对特定文件的读和写权限
5. 维护一张用户和对应文件关系的列表

server模块：核心模块，处理FTP连接以及数据传输

主要功能：

1. 不断更新服务器状态建立新连接
2. 注册常用的FTP命令
3. 不断更新连接实例状态
4. 发送响应以及发送数据到客户端
5. 服务端接收数据

6. 进行相关指令的处理解析，相关解析内容见下文

2.4 功能实现

我们使用仓颉编程语言实现了FTP常见的几个命令工具，`ls`、`pwd`、`cd`、`get`以及`put`，分别对应不同的FTP命令，目前该简易FTP服务器采用主动连接的工作方式并支持以下FTP常用指令：

- USER：输入用户名
- PASS：输入用户密码
- QUIT：退出用户登录
- PORT：指定数据连接端口
- NLST：列出当前目录下的文件（Windows上FTP客户端适用）
- LIST：列出当前目录下的文件（UNIX系统中使用）
- XPWD：显示当前工作路径（Windows上客户端适用）
- STOR：上传文件
- CWD：更改服务上的工作目录
- RETR：下载文件
- 更多常用命令持续添加中.....

运行FTP服务器

项目根目录下已经提供脚本安装`cangjie`工具链，请注意使用`source run-ftp.sh`运行脚本，否则将导致环境变量设置无法生效。

```
#!/bin/bash

# 定义下载地址和文件名
DOWNLOAD_URL="https://cangjie-lang.cn/v1/files/auth/downLoad?nsId=142267&fileName=Cangjie-0.53.13-linux_x64.tar.gz&objectKey=6719f1eb3af6947e3c6af327"
FILE_NAME="Cangjie-0.53.13-linux_x64.tar.gz"

# 检查 cangjie 工具链是否已安装

echo "确保 cangjie 工具链已安装..."
if ! command -v cjc -v &> /dev/null
then
    echo "cangjie工具链 未安装，尝试进行安装..."
    # 下载文件
    echo "Downloading Cangjie compiler..."
    curl -L -o "$FILE_NAME" "$DOWNLOAD_URL"
```

```
16
17      # 检查下载是否成功
18      if [ $? -eq 0 ]; then
19          echo "Download completed successfully."
20      else
21          echo "Download failed."
22          exit 1
23      fi
24
25      # 解压文件
26      echo "Extracting $FILE_NAME..."
27      tar -xvf "$FILE_NAME"
28
29      # 检查解压是否成功
30      if [ $? -eq 0 ]; then
31          echo "Extraction completed successfully."
32      else
33          echo "Extraction failed."
34          exit 1
35      fi
36
37      # 检查 envsetup.sh 是否存在并进行 source
38      if [[ -f "cangjie/envsetup.sh" ]]; then
39          echo "envsetup.sh found!"
40          source cangjie/envsetup.sh
41      else
42          echo "envsetup.sh not found!"
43          exit 1
44      fi
45
46  fi
47
48  # 检查 openEuler 防火墙状态
49  echo "检查 openEuler 防火墙状态..."
50  if systemctl status firewalld | grep "active (running)" &> /dev/null; then
51    echo "防火墙已开启，尝试开放 21 端口..."
52    firewall-cmd --zone=public --add-port=21/tcp --permanent
53    firewall-cmd --reload
54    echo "21 端口已开放。"
55  else
56    echo "防火墙未开启，无需开放端口。"
57  fi
58
59  # 切换到 cangjieFTP 目录
60  cd cangjieFTP || { echo "cangjieFTP 目录不存在"; exit 1; }
61
62  # 编译ftp_server
```

```
echo "正在编译 ftp_server..."
cjpm build

# 检查编译是否成功
if [ $? -eq 0 ]; then
    echo "编译成功."
else
    echo "编译失败."
    exit 1
fi

# 运行 ftp_server
echo "正在启动 ftp 服务器..."
cjpm run
```

ls

- `ls` 用以列出所在目录下的文件以及目录。
- 主要是处理FTP标准中的NLST命令，注意涉及到数据传输，需结合PORT指令的解析一并使用。
- 核心代码：

```
// 处理NLST指令，暂未对参数进行解析
private func nlst(args: String) {
    sendResponse(150, "Sending file list...")
    let path = Directory(currentDir)
    let files = path.entryList()
    var data = ""
    // 当前目录下的文件
    for (file in files) {
        data += file.path.fileName.toString()
        data += "\r\n"
    }
    sendData(data.toArray())
    sendResponse(226, "The list was sent...")
}
```

利用仓颉的文件模块，我们即可通过维护的当前路径 `currentDir` 获取当前目录下的文件信息，并将其发送给客户端。

cd

- `cd` 指令帮助我们进行工作目录的切换。
- 主要是处理FTP标准指令中的CWD命令。

- 核心代码：

```
// 处理CWD指令
private func cwd(path: String) {
    // 首先判断是否存在子目录而后判断是否仍为目录
    if (Directory.exists(currentDir.join(path)) &&
currentDir.join(path).isDirectory()) {
        currentDir = currentDir.join(path)
        sendResponse(250, "Working directory changed...")
        return
    }
    sendResponse(550, "Not a valid dictory...")
}
```

由于我们建立连接时便维护一个工作目录，处理CWD指令只需要判断要切换到的目录是否合理，如果合理则将 `currentDir` 更改为新目录即可。由于使用的是仓颉为我们提供的 `Path` 类型，无需我们自己去处理复杂的路径情况，简单易用。

pwd

- `pwd` 用以获知当前所在工作目录。
- 主要处理FTP标准指令中的XPWD。
- 核心代码：

```
// 处理XPWD指令
private func xpwd() {
    sendResponse(257, "CWD NAME: " + currentDir.toString())
}
```

直接向客户端传送之前维护的当前工作目录即可。

get

- `get` 用以获取服务器上的文件到本地。
- 主要处理FTP指令中的RETR指令，注意通用需结合PORT使用。
- 核心代码：

```
// 处理RETR指令，用以从服务器下载文件
private func retr(path: String) {
    if (File.exists(currentDir.join(path))) {
```

```
        // 进行权限验证，判断是否具有读和下载的权限
        if (!server.getAuthenticator().readPerm(username,
currentDir.join(path))) {
            sendResponse(552, "Permission denied...")
            return
        }
        sendResponse(150, "Sending the file...")
        try {
            let file = File.openRead(currentDir.join(path))
            let bytes = file.readToEnd()
            sendData(bytes)
            sendResponse(226, "File send...")
        } catch (e: Exception) {
            sendResponse(999, e.message)
        }
        return
    }
    sendResponse(550, "Not a valid file...")
}
```

主要逻辑就是首先判断文件是否存在，接着权限验证，判断是否具有下载权限。如果权限验证通过则将文件数据传到客户端即可。

put

- `put` 用以上传文件到服务器。
- 主要处理 `STOR` 指令，通用需要PORT的协助。
- 核心代码：

```
// 处理STOR指令，用以上传文件到服务器
private func stor(path: String) {
    if (!server.getAuthenticator().writePerm(username, currentDir)) {
        sendResponse(552, "Permission denied...")
        return
    }
    sendResponse(150, "Receiving a file...")
    if (File.exists(currentDir.join(path))) {
        // 文件存在，进行覆盖操作
        File.delete(currentDir.join(path))
    }
    File.create(currentDir.join(path))
    try {
        receiveData(currentDir.join(path))
        sendResponse(226, "file received...")
```

```
16        } catch (e: Exception) {
17            sendResponse(999, e.message)
18        }
19    }
```

基本逻辑为首先判断是否当前用户对当前目录具有上传权限，权限认证通过则从客户端读取文件数据从而实现文件上传功能。

3. 环境配置

- 查看openEuler是否符合仓颉工具链的系统环境要求

```
[root@openeuler ~]# ldd --version
ldd (GNU libc) 2.28
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
[root@openeuler ~]# uname -r
4.19.90-2003.4.0.0036.oe1.x86_64
[root@openeuler ~]# rpm -qa | grep libstdc++
libstdc++-7.3.0-20190804.h31.oe1.x86_64
[root@openeuler ~]#
```

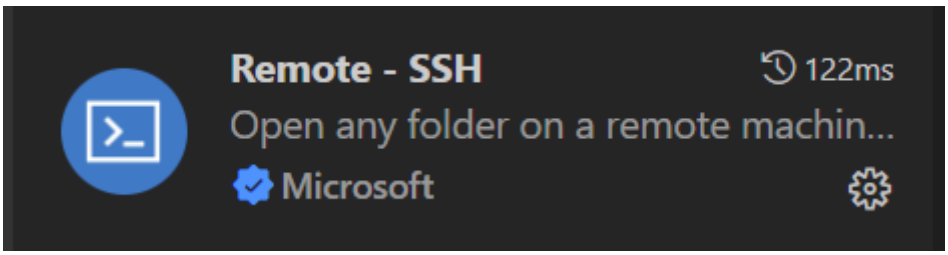
- 安装仓颉工具链，openEuler似乎并不需要安装仓颉工具链依赖。
  - 可使用提供的脚本一键安装。
- 运行第一个仓颉程序

```
[root@openeuler cangjieCode]# vim hello.cj
[root@openeuler cangjieCode]# cjc hello.cj -o hello
[root@openeuler cangjieCode]# ls
default.bchir2  default.cjo  hello  hello.cj
[root@openeuler cangjieCode]# hello
-bash: hello: command not found
[root@openeuler cangjieCode]# ./hello
你好，仓颉
```

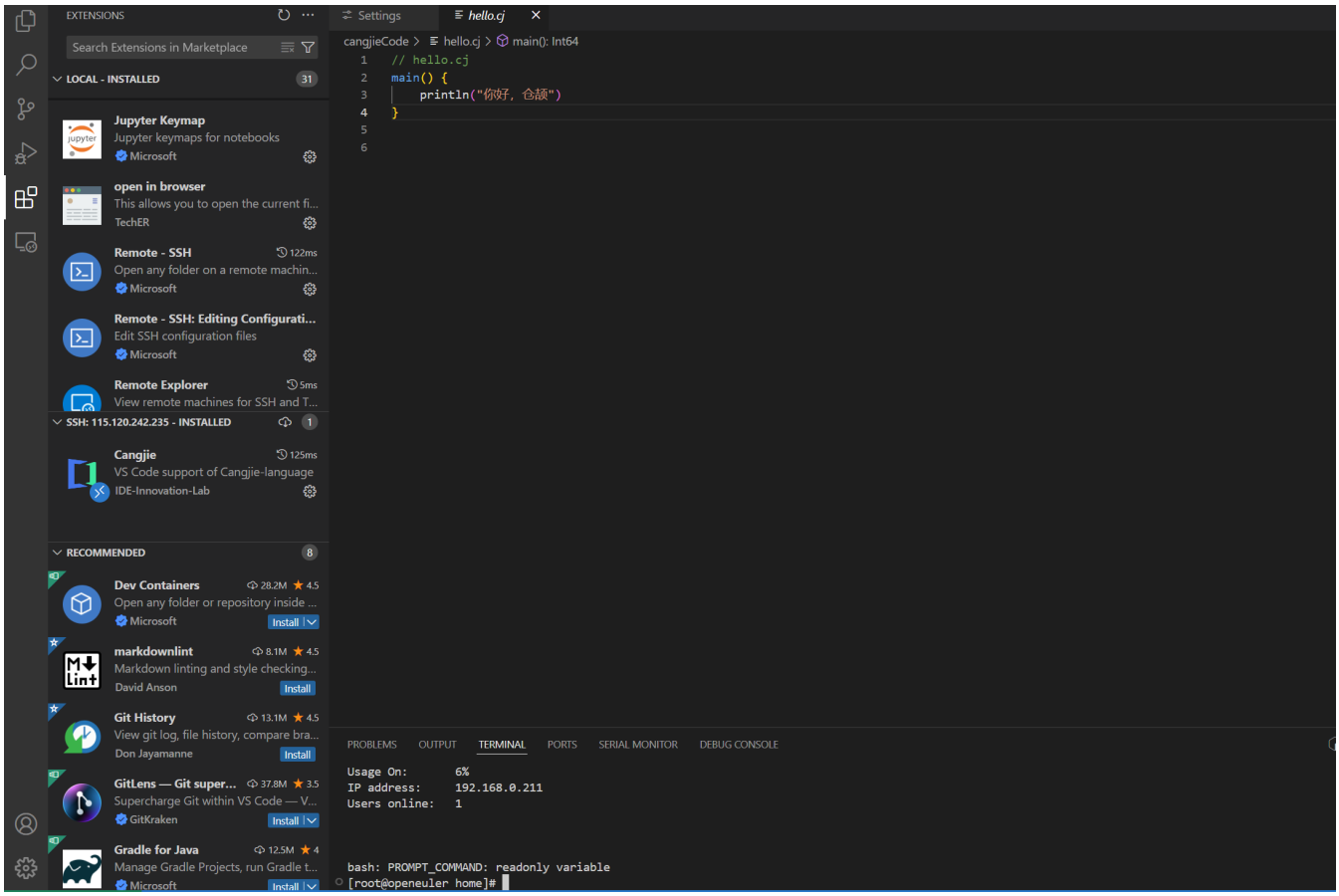
- 使用vscode搭建编码环境
  - 修改 `/etc/ssh/sshd_config` 文件中的 `AllowTcpForward` 为 `yes`

```
#CheckUserSplash yes
PermitRootLogin yes
PasswordAuthentication yes
UseDNS no
Protocol 2
LogLevel VERBOSE
PubkeyAuthentication yes
RSAAuthentication yes
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
PermitEmptyPasswords no
PermitUserEnvironment no
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com
ClientAliveCountMax 0
Banner /etc/issue.net
MACs hmac-sha2-512,hmac-sha2-512-etm@openssh.com,hmac-sha2-256,hmac-sha2-256-etm@openssh.com,hmac-sha1,hmac-sha1-etm@openssh.com
StrictModes yes
AllowTcpForwarding yes
AllowAgentForwarding no
GatewayPorts no
PermitTunnel no
KexAlgorithms curve25519-sha256,curve25519-sha256@libssh.org,diffie-hellman-group14-sha1,diffie-hellman-group-exchange-sha1,diffie-hellman-group-exchange-sha256
```

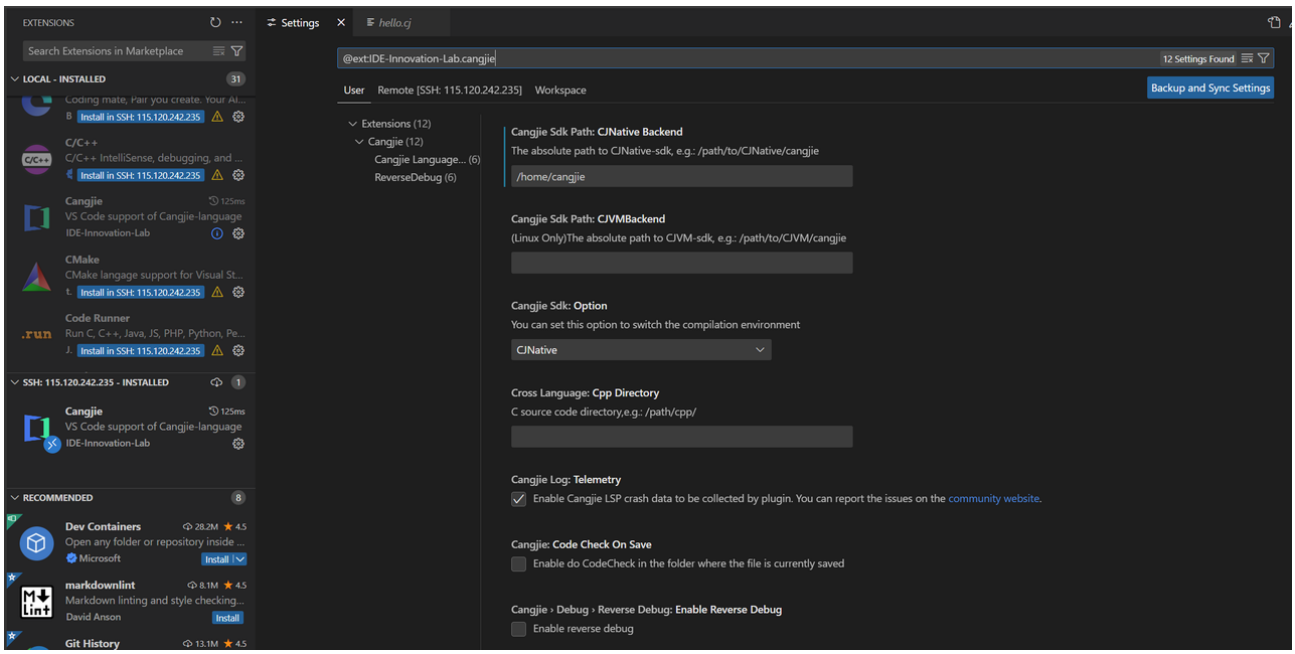
- 本地vscode安装相关远程连接插件



- 进行远程连接连接到远端的openEuler操作系统



- 为了更好的使用仓颉语言，还需安装仓颉插件
  - https://cangjie-lang.cn/download/0.53.13
  - 在上述安装地址中下载VScode Plugin并将其上传到openEuler系统，解压后通过VScode安装本地插件并配置插件的SDK的路径为此前下载解压的仓颉的路径



- 至此，我们在openEuler操作系统上搭建Cangjie语言的开发环境就已经完成。

4. 测试报告

4.1 用户界面测试

测试目标：

验证FTP服务器的命令行界面布局、交互元素和响应是否符合设计要求。

测试内容及结果：

- **布局测试**：命令行界面布局合理，所有提示信息正确显示，无错位或重叠现象。
- **交互测试**：命令行输入响应正确，如 `ls`，`cd`，`put`，`get` 等命令能够正确执行，无卡顿或失效现象。
- **响应测试**：命令执行速度快，无明显的延迟或错误响应。

结论：

用户界面测试通过，符合设计要求。

4.2 可靠性测试

测试目标：

验证FTP服务器在异常情况和长时间运行下的稳定性和容错能力。

测试内容及结果：

- **异常测试**：模拟网络中断、服务器异常重启等异常情况，FTP服务器能够给出友好提示并保持稳定运行。
- **长时间运行测试**：FTP服务器连续运行 72 小时，未发现内存泄漏、性能下降或崩溃现象。

结论：

可靠性测试通过，FTP服务器具有较高的稳定性和容错能力。

4.3 易用性测试

测试目标：

评估FTP服务器的易用性和用户体验。

测试内容及结果：

- **操作测试**：用户能够顺利完成文件上传、下载、列表查看等操作，操作流程符合用户习惯。

结论：

易用性测试通过，FTP服务器用户体验良好。

4.4 安全性测试

测试目标：

验证FTP服务器的安全性，包括用户认证、权限控制的安全性。

测试内容及结果：

- **用户认证测试**：用户登录时，只有正确的用户名和密码才能成功认证，匿名登录权限受限。
- **权限控制测试**：不同用户根据权限能够访问和操作相应的文件和目录，无越权访问现象。

结论：

安全性测试通过，FTP服务器具备基本的安全保障。

4.5 性能测试

测试目标：

评估FTP服务器的文件传输效率。

测试内容及结果：

- **文件传输效率测试**：在不同网络条件下，文件上传和下载速度满足预期，无显著延迟。

结论：

性能测试通过，FTP服务器能够满足预期的文件传输性能要求。