CS232 Operating Systems
Assignment 4: Concurrent File Transfer Application

CS Program
Habib University

Fall 2024
Release Date: 21 November 2024 @ 12:00PM
Due Date: 29 November 2024 @ 11:59PM

Introduction

In today's fast-paced digital world, efficient file transfer protocols are essential for managing large datasets across networks. This assignment will give you hands-on experience in creating a multi-threaded file transfer system. You will design both a server process to handle file requests and a client process to request and reassemble files. The goal is to ensure the file is split, transmitted, and reassembled correctly across concurrent connections while maintaining data integrity.

Rationale

As data transfer needs increase, especially in distributed systems, it's crucial to understand how concurrent file handling works. This assignment introduces multi-threaded programming, an important skill for building efficient client-server applications and network protocols.

Objectives

The main objectives of this assignment are:

- Implement a background server process to handle file transfer requests from multiple clients.
- Use multi-threading to transfer file segments concurrently.
- Ensure the reassembled file is accurate and maintains data integrity.
- Conduct automated tests to verify correctness across different scenarios.

Assignment Requirements and Tasks

You will implement two main components: a server (Process A) and a client (Process B). They should communicate using sockets, as this is a simple and widely-used IPC mechanism for networked file transfers. For initial testing, you can use the local host ip that is 127.0.0.1.

1. **Setup and Server Functionality:**
    - Process A should act as a background server, continuously listening for incoming file transfer requests from Process B.
    - Upon receiving a request, Process A should split the requested file into segments and transfer each segment in a separate concurrent session (thread).

2. **Client Functionality:**
    - Process B should initiate a request to Process A to transfer a file, specifying the file name and the number of concurrent sessions required using command line arguments.
    - Upon receiving the segments, Process B should reassemble the file parts to recreate the complete original file.
3. **File Transfer and Integrity:**
    - Implement data integrity checks to verify that all file parts are received and assembled correctly. You can implement checksums or hashes (e.g., MD5, SHA256) to ensure data integrity during transfer. For this you can access the file locally to extract the checksum or hash and then compare it to the checksum or hash created by the consolidated file from the server.
    - Test the application with varied file types, such as text files, images, videos, and compressed files, as well as different numbers of threads.
4. **Automated Testing:**
    - Develop automated tests to verify correctness across various scenarios, including edge cases.
    - Include 9 test cases of varied complexity, with different file types, sizes, and concurrent session numbers.

Test Data and Cases

For evaluation, use a variety of files, including:

- Simple text files (1 KB to 10 MB)
- Image files (10 KB to 2 MB)
- Video files (1 MB to 200 MB)
- Compressed files (ZIP)

Automated tests should include the following scenarios:

- Varying thread counts (e.g., 1, 5, 10, 20)
- Different file types and sizes
- Corrupted file handling

Submission Instructions

Submit a single compressed zip file named `CSxxyyyy.zip` (replace with your student ID) via the Assignment 4 module on LMS by 11:59 PM on November 30, 2024. The zip file must contain:

1. A `makefile` with `compile`, `build`, and `clean` targets.
2. All relevant .c source files that is `server.c` and `client.c` for the assignment.
3. Tests`.sh` files to automate the testcases evaluation.
4. A short README detailing how to run the server and client processes, including any assumptions or limitations.

Using chatGPT or any other AI Tool

You are not permitted to use any AI tool to obtain code for this assignment. The faculty may conduct a viva and use AI-detection tools to verify originality. If AI tools like ChatGPT are used, you must provide the chat history. Failure to comply or using AI-generated content without adaptation may result in a zero (0) and referral to the Office of Academic Conduct. This assignment is to be completed independently, and originality will be strictly enforced. You may refer to online resources but must not directly copy code.

Plagiarism Policy

We have zero tolerance for plagiarism. Every submission will be screened using a plagiarism detection software. Offenders will be reported to the Office of Academic Conduct and obtain zero (0). This applies even if the code is obtained from an online repository like github without proper attribution. We expect you to credit all sources used for completion of this assignment.

Rubric

The following rubric will be used to evaluate your submission:

Criteria	**Points**	**Description**
Correct Functionality	40	Server and client perform file transfer correctly; file parts are merged accurately at the client.
Multi-threading Implementation	20	Server uses multi-threading effectively; concurrency is handled correctly with varying session counts.
Automated Testing	20	Tests cover a wide range of cases as mentioned in the test case mapping given in Table 1 and verify correctness under various conditions.
Data Integrity	10	File integrity is maintained; the file received matches the original.
Code Quality and Comments	10	Code is well-organized, readable, and thoroughly commented.

Table 1: Test Case Overview

Test Case	File Type	Size	Number of Threads
Test 1	Simple Text	1 KB	1
Test 2	Simple Text	1 KB	5
Test 3	Large Text	10 MB	10
Test 4	Small Image	50 KB	5
Test 5	Large Image	500 KB	10
Test 6	Small Video	1 MB	10
Test 7	Moderate Video	10 MB	20
Test 8	Compressed Zip	5 MB	5
Test 9	Corrupted File	10 KB	5

Penalties

You will be penalized for the following:

(a) submission does not follow the given instructions that is you have submitted files which are not required (for e.g. you are submitting test case evaluation data files) or have failed to comply to the instructions: -50% marks.

(b) code does not compile: -20% marks.

(c) code has warnings (compile with -Wall): -10% marks.

(d) makefile is absent: -10% marks.

(e) program does not work properly or crashes: -20% marks

(f) late submission: -10% marks for missing the deadline + -5% $\times$ num. of days.