Large Efficient Flexible and Trusty

Name: Aimi

StudentID:

- 2022年8月5日



Abstract

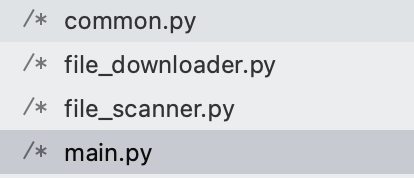
The main purpose of this project is to achieve mutual synchronization of large files. In the transmission process, the upper-layer control protocol implemented by itself is used, and the socket is changed to make it synchronize data faster. Breakpoint transmission and modification retransmission are also used to make the project more complete and flexible. The application layer protocol is transmitted based on TCP.

Introduction

**background**：**File** sharing is a commonly used network-based application in our daily life. There are a lot of excellent apps that provided such services, such as Dropbox, Google Drive, Baidu NetDisk, iCloud.

**Project Aim**: This project aims at using Python Socket network programming to design and implement a Large Efficient Flexible and Trusty (LEFT) Files Sharing program. From the name of this project.

Methodology

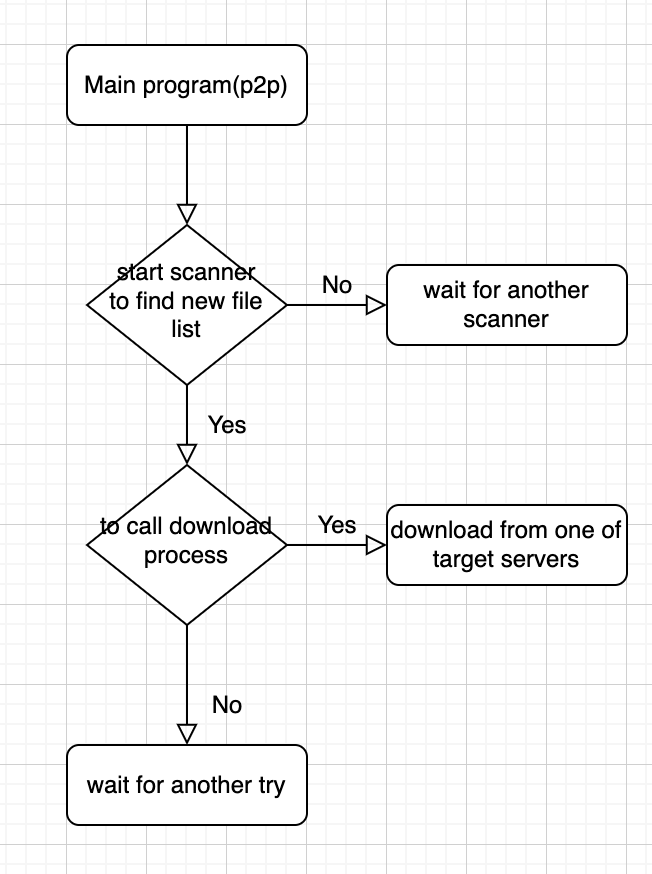


We first perform a file scan on the share folder, then sort all files by timestamp (descending order): find the most recently updated file list, and then each program running on a different PC maintains a dictionary of its own.

In the dictionary, we maintain each file name and the corresponding file size (there will be more time to expand the project in the future, we can also maintain the md5 check, which will be more accurate). Through the file name and size and timestamp, we take out the local dictionary and communicate with the remote server (each ip) and compare the difference.

If it is found that there are dictionary items that do not exist locally and exist in other network nodes, we call the downloader project to inquire and download (if it is multi-node, it will be downloaded in the order of the server list)

Implementation



**My steps:**

1. Course study requirements.

2. Begin to divide the corresponding function into modules, write down the function name and the function of this function.

3. Improve the function module. Functions that can be grouped together form a class, and objects are used to perform functions.

4. Form reusable functions in functional modules, and try to make each function only do one thing.

5. Clear the context of the function module and start writing multi-process.

6. Test and Improve

**Main skill used:**

Object-oriented programming, modular and multithreading are used in this coursework.

Object-Oriented In Python

Process-oriented programming treats a computer program as a series of commands, that is, the sequential execution of a set of functions. In order to simplify the program design, the process-oriented function is further divided into sub-functions, that is, the large function is divided into small functions to reduce the complexity of the system.

In object-oriented programming, a computer program is regarded as a collection of objects, and each object can receive messages from other objects and process these messages. The execution of a computer program is a series of messages passed between objects. .

In Python, all data types can be treated as objects, and of course custom objects are also possible. A custom object data type is the concept of a class in object-orientedd.

**Multithreading**

The multiprocessing module is used to do parallel operations. There is a Global Interpretation Lock (GIL) inside Python. Only one thread of any process is allowed to perform CPU operations at any time. If a thread in a process obtains the GIL while performing CPU operations, other threads will not be able to perform CPU operations and can only wait. , so that multi-threading cannot take advantage of the multi-core feature of the CPU. Multiprocessing actually generates an operating system process for each task, and each process is individually assigned to the Python interpreter and GIL, so the program has multiple GILs in operation, and each runner's thread Will get a GIL, run in different environments, and naturally can be assigned to different processors

**Python Modular**

Now the idea of modular programming. Modular programming helps developers to coordinate, divide and collaborate, and improve code flexibility and maintainability. For example, many engineers work together to develop the same system. A is the logic of the login function, B is the logic of the registration function, and C is the logic of the user management. These codes are separated, and they are inherited into the same system through modular assembly.

We divide into:

common.py

file\_downloader.py

file\_scanner.py

main.py

Testing and Result

We test in many situations:

3 servers situations:

python3 main.py --ip 192.168.0.1,192.168.0.2

2 servers situations:

on osx:

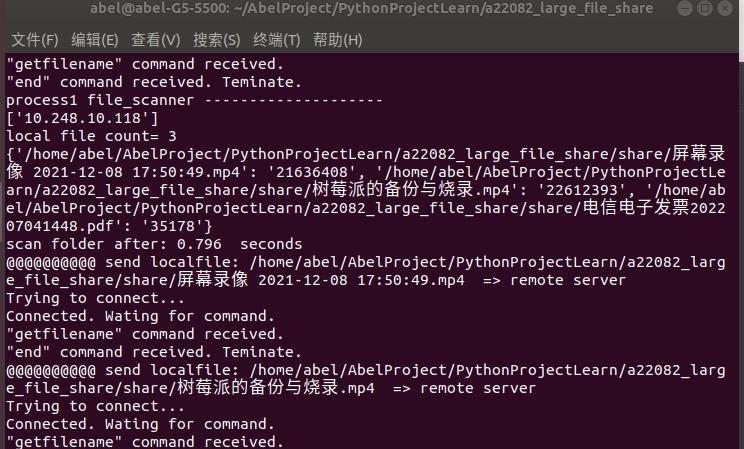
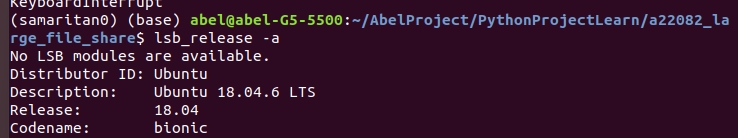
python3 main.py --ip 10.248.10.129

on another ubuntu server:

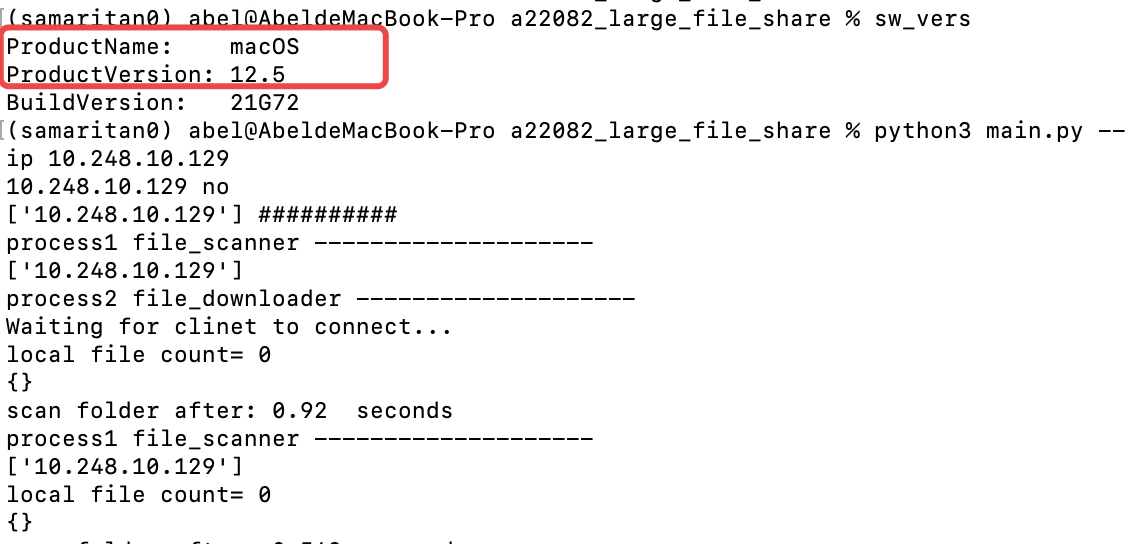
python3 main.py --ip 10.248.33.229

The screenshots are 2 servers situation: I shot the working status as following:

On Ubuntu side:



ON Mac :





Conclusion and reference

In this Porject, I designed the application layer protocol for file synchronization, designed the system for file synchronization, and implemented it in `python`. This system can transfer files to and from different devices with encryption, compression, breakpoint transfer, modification and retransmission, and it passed my test.

References:

<https://docs.python.org/zh-cn/3/library/multiprocessing.html>

https://zh.wikipedia.org/wiki/%E6%96%87%E4%BB%B6%E4%BC%A0%E8%BE%93%E5%8D%8F%E8%AE%AE