1. Sumário do Problema a Ser Tratado

O problema consiste em desenvolver um sistema que permita a distribuição de arquivos enviados por clientes para múltiplos servidores de processamento, utilizando um portal intermediário. O portal deve ser capaz de distribuir os arquivos de acordo com dois modos: round-robin e aleatório, assegurando a correta entrega dos arquivos e recebimento das respostas dos servidores.

2. Descrição dos Algoritmos e Tipos Abstratos de Dados

Algoritmos:

- **Round-Robin**: Distribuição cíclica dos arquivos enviados para os servidores.
- **Aleatório**: Distribuição aleatória dos arquivos para os servidores.

Tipos Abstratos de Dados:

- **Listas**: Para armazenar os servidores disponíveis e suas portas.
- **Strings**: Para manipulação e envio de comandos e arquivos.

Principais Funções e Procedimentos:

Portal (portal.py):

- **escolher_servidor()**: Função que escolhe um servidor com base no modo de seleção especificado (round-robin ou aleatório).
- **lidar_com_cliente(cliente_socket)**: Função que lida com as conexões dos clientes, recebendo arquivos e distribuindo-os para os servidores apropriados.
- **main()**: Função principal que inicializa o socket do portal e aguarda conexões dos clientes.

Cliente (cliente.py):

- **enviar_arquivos_portal(arquivos, portal_ip, portal_porta)**: Função que envia arquivos listados pelo cliente para o portal.
- **listar_arquivos()**: Função que lista arquivos .c no diretório atual (implementação local).

Servidor (servidor.py):

lidar_com_pedido(servidor_socket): Função que lida com as conexões dos clientes (portal), recebendo arquivos, compilando e executando-os, e enviando de volta os resultados.

3. Decisões de Implementação Omissas na Especificação

Tratamento de Erros:

- O código inclui tratamento de exceções para capturar erros durante a conexão, envio de arquivos e compilação. Isso garante que o sistema possa lidar com falhas sem interromper a operação global.

Threading:

- A utilização de threading no portal permite que múltiplos clientes sejam atendidos simultaneamente, melhorando a eficiência do sistema.

4. Retransmissão de Mensagens

A retransmissão de mensagens é tratada com o protocolo TCP, que garante a entrega confiável dos dados. Em caso de falha na transmissão, o TCP reenvia os pacotes de dados automaticamente até que a entrega seja confirmada, não necessitando de implementação adicional de retransmissão no código.

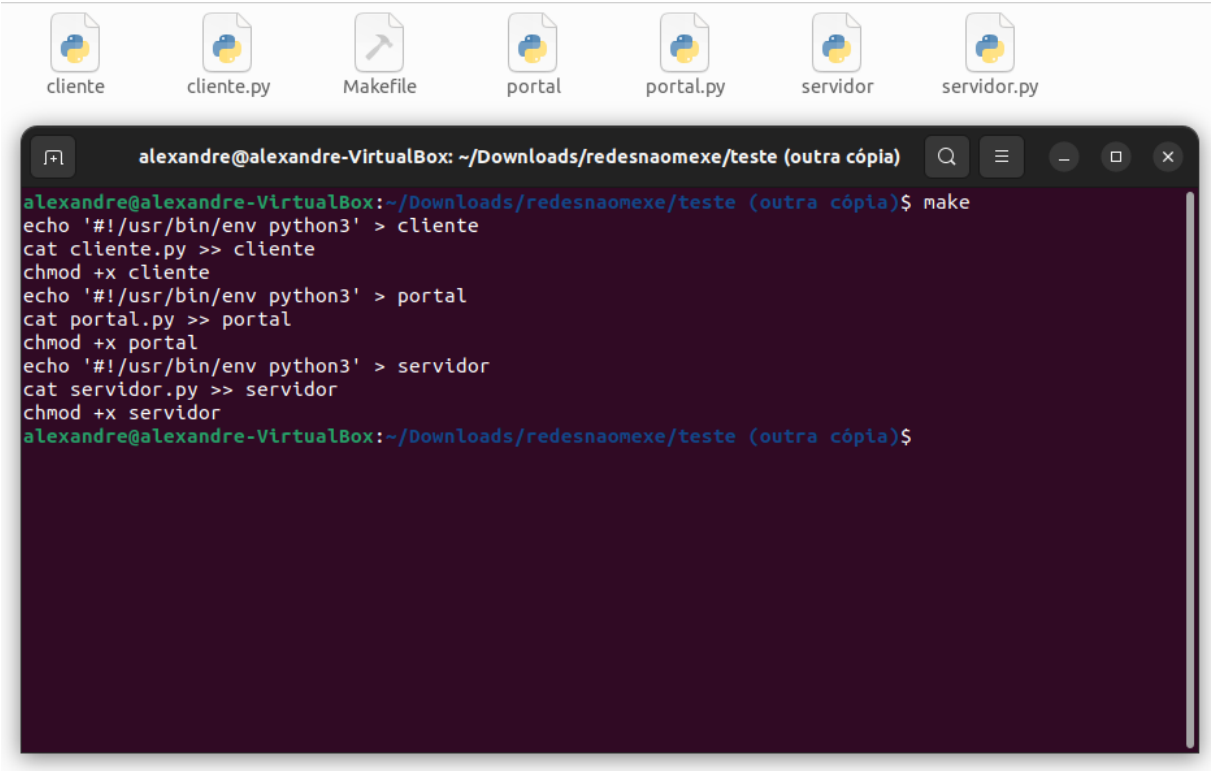
5. Testes e Resultados

Configuração do Ambiente de Testes:

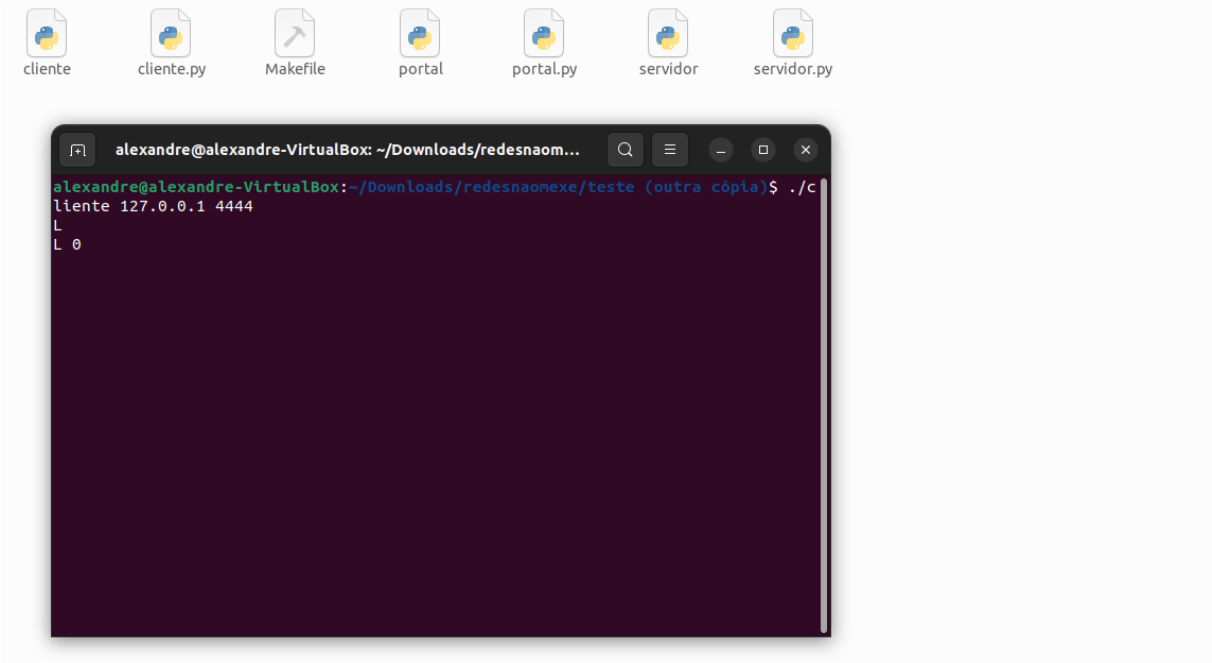
Os testes foram realizados em uma rede local, com componentes executados na mesma máquina.

Cenários de Teste:

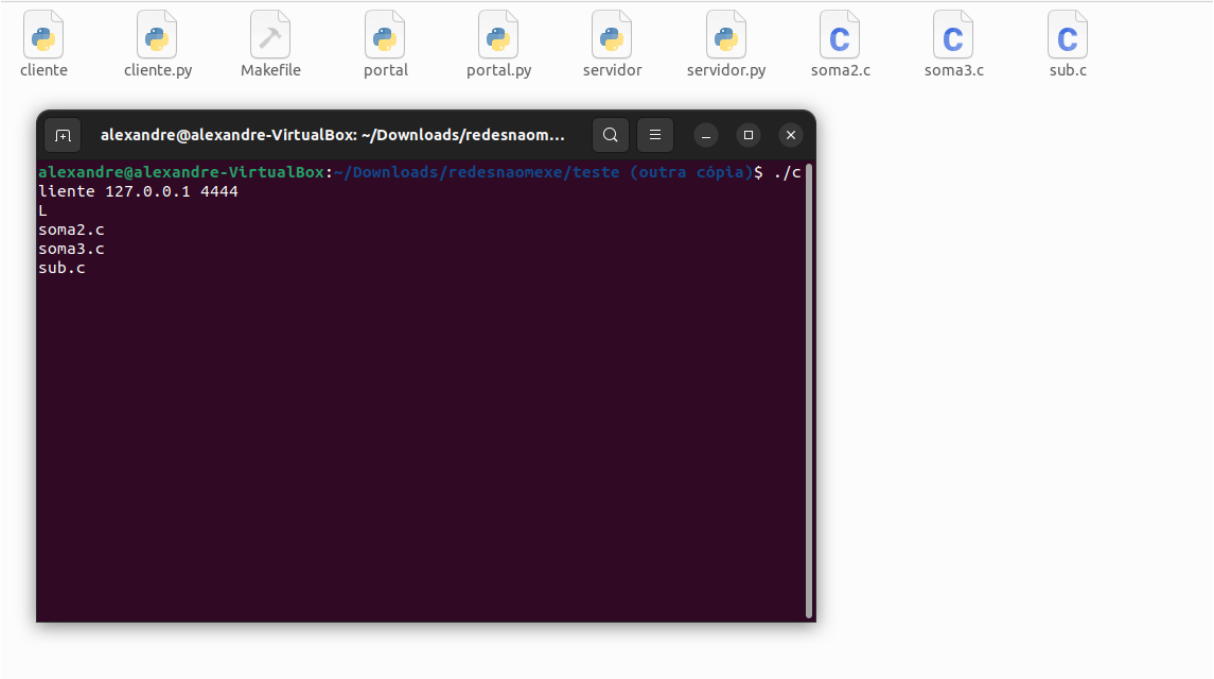
- **Round-Robin**: Vários arquivos foram enviados sequencialmente, e a ordem de distribuição seguiu o ciclo entre os servidores.
- **Aleatório**: Arquivos foram enviados e distribuídos aleatoriamente, confirmando a aleatoriedade.



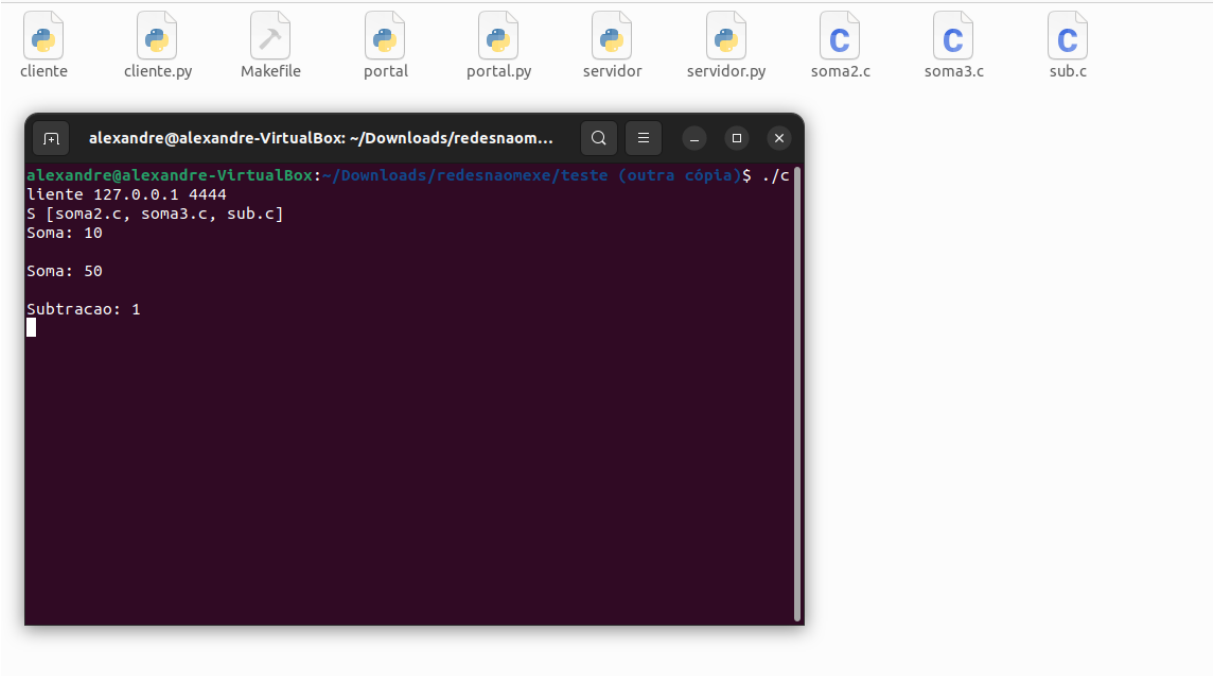
- Execução do Make.



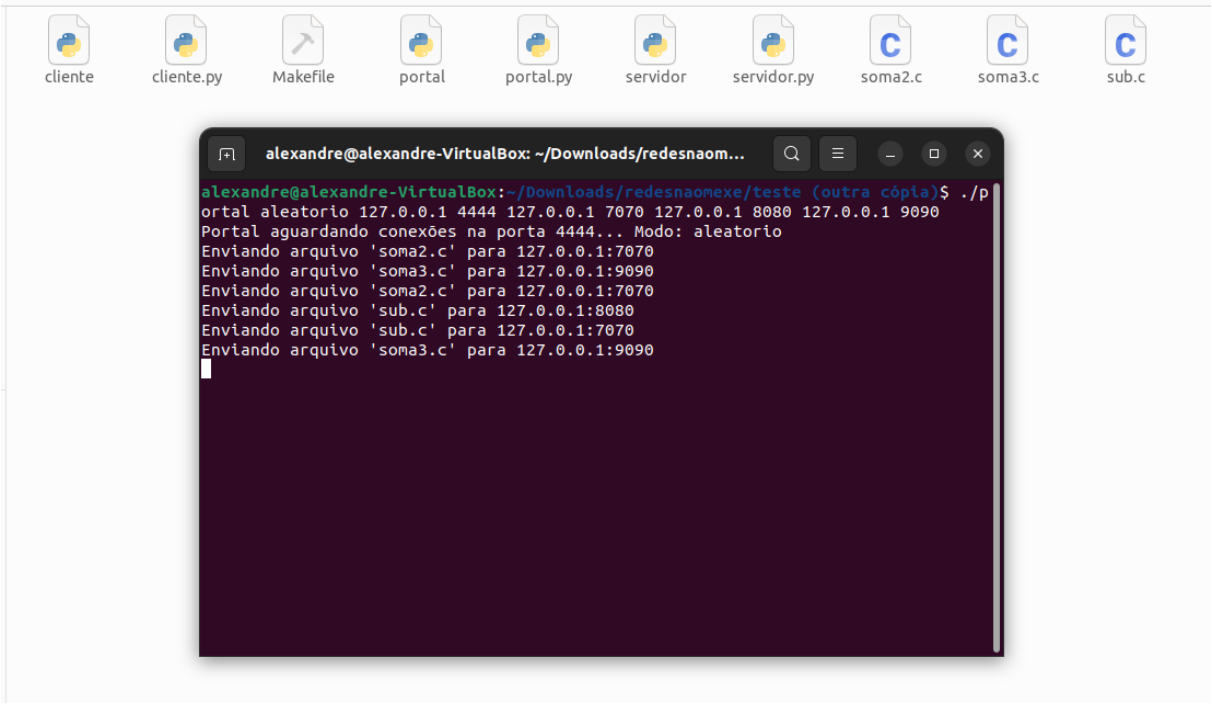
- Comando “L” quando não há arquivos no diretório.



- Comando “L” quando há arquivos presentes no diretório.



- Execução do envio de arquivos e retorno de reposta ao cliente.



- Função no PORTAL para testes de Round-Robin e Aleatório (No caso da imagem está operando no modo aleatório). Ele mostra o qual o arquivo e para qual servidor ele está sendo enviado.

Conclusão

O sistema desenvolvido atende aos requisitos especificados, proporcionando uma solução eficiente para a distribuição de arquivos de .C para múltiplos servidores de processamento. A implementação permite que múltiplos clientes enviem arquivos simultaneamente, com o portal distribuindo esses arquivos de forma equilibrada entre os servidores, utilizando modos de seleção como round-robin e aleatório.

Referências

- Documentação das bibliotecas padrão do Python: Python Standard Library
- Artigos e tutoriais sobre comunicação TCP e programação de sockets em Python.