Documentação do Sistema de Gerenciamento de Arquivos Cliente-Servidor

Visão Geral
Este sistema permite o gerenciamento remoto de arquivos entre um cliente e um servidor através de uma conexão de rede, implementando operações básicas como upload, download, listagem e exclusão de arquivos.

Arquivo client.c

Estrutura e Componentes

1. Configuração Inicial
   - set_console_encoding(): Configura o terminal para suportar caracteres especiais e acentos
   - Definições de constantes (`PORT`, `BUFFER_SIZE`) para configuração da conexão

2. Funções Auxiliares
   - show_progress(): Exibe uma barra de progresso visual durante transferências
   - show_complete_message(): Mostra mensagens de conclusão de operações
   - list_local_files(): Lista arquivos do diretório local do cliente
   - `select_file_from_list()`: Interface para seleção de arquivos locais
   - `get_download_path()`: Permite ao usuário especificar o destino de downloads

3. Lógica Principal
   - Inicializa a conexão Winsock
   - Estabelece conexão com o servidor
   - Menu interativo com 6 opções:
     1. LIST - Listar arquivos no servidor
     2. UPLOAD - Enviar arquivo para o servidor
     3. DOWNLOAD - Baixar arquivo do servidor
     4. DELETE - Excluir arquivo no servidor
     5. LOCAL - Listar arquivos locais
     6. EXIT - Sair do programa

Fluxo de Operações

1. Upload:
   - Lista arquivos locais
   - Seleciona arquivo para envio
   - Exibe barra de progresso durante transferência
   - Mostra confirmação do servidor

2. Download:
   - Lista arquivos disponíveis no servidor
   - Seleciona arquivo para download
   - Especifica diretório de destino
   - Exibe progresso e confirma recebimento

3. Delete:
   - Lista arquivos no servidor
   - Seleciona arquivo para exclusão
   - Mostra barra de progresso simulada
   - Exibe confirmação da operação

Arquivo server.c

Estrutura e Componentes

1. Configuração Inicial
   - Mesma configuração de encoding do cliente
   - Definição do diretório de armazenamento (SERVER_STORAGE)

2. Funções Principais
   - create_storage_directory(): Cria o diretório de armazenamento se não existir
   - list_files(): Lista arquivos disponíveis no servidor
   - upload_file(): Recebe e armazena arquivos enviados pelo cliente
   - download_file(): Envia arquivos solicitados pelo cliente
   - delete_file(): Remove arquivos do servidor

3. Lógica do Servidor
   - Inicializa Winsock e cria socket
   - Configura endereço e porta
   - Aguarda conexões de clientes
   - Processa comandos recebidos:
     - LIST: Retorna lista de arquivos
     - UPLOAD: Recebe e armazena arquivo
     - DOWNLOAD: Envia arquivo solicitado
     - DELETE: Remove arquivo especificado
     - EXIT: Encerra conexão com cliente

Fluxo de Operações

1. Inicialização:
   - Cria diretório de armazenamento
   - Abre socket e aguarda conexões

2. Conexão com Cliente:
   - Aceita nova conexão
   - Entra em loop de processamento de comandos

3. Processamento:
   - Para cada comando recebido, executa a função correspondente
   - Envia respostas e confirmações ao cliente
   - Mantém registro das operações no console

Protocolo de Comunicação

1. Formato dos Comandos:
   - LIST: Sem parâmetros
   - UPLOAD [nome_arquivo]
   - DOWNLOAD [nome_arquivo]
   - DELETE [nome_arquivo]
   - EXIT: Encerra conexão

2. Fluxo de Dados:
   - Cliente envia comando como string
   - Servidor processa e retorna resultado
   - Para transferências de arquivos:
     - Cliente/servidor envia dados em blocos
     - Barra de progresso é atualizada
     - Confirmação é enviada ao final

Compilação e Execução

1. Compilar:
   ```bash
   gcc server.c -o server.exe -lws2_32
   gcc client.c -o client.exe -lws2_32
   ```

2. Executar:
   - Primeiro inicie o servidor:
     ```bash
     server.exe
     ```
   - Depois inicie o cliente em outro terminal:
     ```bash
     client.exe
     ```

Considerações Finais

- O sistema foi projetado para Windows, utilizando a API Winsock
- Todas as operações incluem feedback visual para o usuário
- O código está preparado para lidar com caracteres especiais e acentos
- O servidor suporta múltiplas conexões sequenciais (uma por vez)

Melhorias Futuras

1. Implementar transferência de metadados (tamanho real do arquivo antes do download)
2. Adicionar autenticação de usuários
3. Implementar transferência de múltiplos arquivos simultaneamente
4. Adicionar suporte para diretórios recursivos
5. Implementar sistema de logs mais robusto