MyFTP - Cliente e Servidor FTP Simples

O objetivo desse projeto é desenvolver uma aplicação cliente e servidor que implementa uma versão simplificada do protocolo **FTP**. As duas componentes principais do módulo são os executáveis do cliente `myftp` e do servidor `myftpserver`. Os comandos FTP implementados são:

1. **get** (`get <arquivo_remoto>`) - Copia o arquivo com o nome <arquivo_remoto> do diretório remoto para o diretório local.
2. **put** (`put <arquivo_local>`) - Copia o arquivo com o nome <arquivo_local> do diretório local para o diretório remoto.
3. **delete** (`delete <arquivo_remoto>`) - Deleta o arquivo com o nome <arquivo_remoto> do diretório remoto.
4. **ls** (`ls`) - Lista os arquivos e subdiretórios no diretório remoto.
5. **cd** (`cd <diretorio_remoto>` ou `cd ..`) - Muda para o <diretorio_remoto> na máquina remota ou muda para o diretório pai do diretório atual.
6. **mkdir** (`mkdir <diretorio_remoto>`) - Cria um diretório com nome <diretorio_remoto> como um subdiretório no diretório de trabalho atual na máquina remota.
7. **pwd** (`pwd`) - Imprime na tela o diretório de trabalho atual da máquina remota.
8. **quit** (`quit`) - Termina a sessão FTP.

Executando a aplicação

Clone o repositório e instale os pacotes necessários. É recomendável criar antes um ambiente virtual no python. No linux, basta executar:

```
$ python3 -m venv <nome> # Cria o ambiente virtual.
$ source <nome>/bin/activate # Ativa o ambiente virtual.
$ pip install -r requirements.txt # Instala os pacotes necessários.
```

Para iniciar o programa servidor execute:

```
$ python myftpserver.py <port_number>
```

Para iniciar programa cliente execute:

```
$ python myftp.py <server_name> <port_number>
```

Especificação da aplicação

O funcionamento do cliente e servidor são explicados a seguir:

- Servidor FTP (programa `myftpserver`) - O programa servidor recebe um único parâmetro de linha de comando que é o número da porta a qual o servidor irá executar. Uma vez que o programa `myftpserver` for invocado, ele cria um *socket* TCP, associa o número da porta do servidor ao seu *socket* e passa a escutar conexões de clientes. Quando uma conexão com um cliente é estabelecida, o servidor começa a aceitar comandos e executá-los. Mensagens de erro apropriadas são enviadas ao cliente sempre que os comandos falham. Ao receber o comando `quit`, o servidor encerra a conexão e fica pronto para aceitar outras conexões.

- Cliente FTP (programa `myftp`) - O programa do cliente FTP recebe dois parâmetros de linha de comando: o endereço da máquina que o servidor reside e o número da porta. Uma vez que o cliente começa a rodar ele exibe um *prompt* `myftp>`. A partir daí aceita e executa comandos enviando-os para o servidor e exibindo os resultados e mensagens de erro quando apropriado. O cliente deve sessar sua execução quando o usuário entrar o comando `quit`.

Implementação do protocolo

O protocolo da camada de aplicação proposto segue o seguinte formato (`[]` indica espaço vazio e `[_]` uma quebra de linha):

Mensagem de requisição

```
COMMAND[]PARAMETER[_]
ClientName[]CLIENT_NAME[_]
FileSize[]FILESIZE[_]
[_]
CONTENT
```

Mensagem de resposta

```
STATUS_CODE[]MESSAGE[_]
[_]
CONTENT
```

Observações **importantes**:

1. O servidor implmentado é concorrente (*multi-threaded*).
2. Assumimos que o usuário sempre digita um comando com a sintaxe correta (i.e. não existe checagem da sintaxe).

Referências

A principal inspiração para esse trabalho foi o material encontrado em Programming-Project1 da Universidade da Geórgia, a qual deixamos os devidos créditos.