Lista De Exercícios #03: SOCKETS - Cliente HTTP

Neste exercício, você deverá criar um programa que recebe uma URL completa, extrai informações relevantes sobre o endereço e, em seguida, estabelece uma conexão via SOCKET para baixar o conteúdo da página ou arquivo. O programa também deve salvar as informações do HEADER da resposta em um arquivo à parte.

Objetivos:

1. Analisar a URL informada:
   1.1. O programa deve permitir que o usuário insira uma URL completa.
   1.2. Extrair as seguintes partes da URL:
      1.2.1. **Host**: domínio ou endereço IP do servidor.
      1.2.2. **Caminho**: o caminho do recurso solicitado (caso exista).
      1.2.3. **Nome do arquivo**: nome do arquivo solicitado (caso a URL contenha um arquivo, como `index.html` ou `image.png`).
2. Detectar a porta a ser utilizada (80 ou 443):
   2.1. A partir do HEADER da URL, o programa deve identificar se a conexão deve ser feita via porta 80 (HTTP) ou porta 443 (HTTPS). Isso pode ser determinado pelo esquema http ou https presente na URL.
3. Estabelecer uma conexão SOCKET:
   3.1. Criar um SOCKET para se conectar ao servidor na porta correta (80 ou 443), usando a biblioteca padrão de sockets.
   3.2. Enviar a requisição HTTP para o servidor, com o método GET.
   3.3. A requisição deve incluir o Host e, se houver, o Caminho e o Nome do arquivo.
4. Gerenciar o Download do Conteúdo:
   4.1. Após a requisição, o programa deve analisar o HEADER da resposta e verificar os campos `Content-Length` ou `Transfer-Encoding` para determinar como o conteúdo será baixado.
      4.1.1. Se o campo `Content-Length` estiver presente, o programa deverá saber o tamanho exato do conteúdo e fazer o download de uma vez só, salvando-o de acordo com o tipo de conteúdo (arquivo ou HTML).
      4.1.2. Caso o campo `Transfer-Encoding: chunked` seja encontrado, o programa deverá lidar com a transferência de dados em partes (chunks). O conteúdo será recebido em pedaços e deverá ser reconstituído corretamente.

5. Salvar o conteúdo da resposta:
   5.1. Se o conteúdo for um arquivo (como imagens, documentos ou outros formatos binários), o programa deve salvar no formato apropriado com o nome do arquivo.
   5.2. Se o conteúdo for uma página HTML, o programa deve salvar o conteúdo em um arquivo `.html`.
6. Salvar o HEADER em um arquivo separado:
   6.1. O HEADER da resposta HTTP deve ser armazenado em um arquivo de texto, para que o usuário possa analisar as informações da resposta.

Requisitos:

1. O programa deve funcionar tanto para URLs com http quanto https.
2. O HEADER da resposta deve ser salvo em um arquivo de texto.
3. O conteúdo deve ser salvo de acordo com seu tipo (arquivo ou HTML).
4. O programa deve lidar com a detecção da porta (80 ou 443) de forma automática a partir da URL.
5. O download do conteúdo deve ser gerenciado de acordo com a presença de `Content-Length` ou `Transfer-Encoding` no HEADER da resposta:
   - Caso o HEADER tenha o campo `Content-Length`, o conteúdo estará no tamanho especificado e o download será gerenciado por esse tamanho. Caso o campo `Transfer-Encoding: chunked` esteja presente, o conteúdo será enviado em pedaços, e o programa deverá gerenciar o processo de leitura de cada pedaço, até que a resposta seja completa.
6. utilizar a biblioteca SOCKETS apenas, utilizando diretamente a conexão e a leitura dos bytes da resposta, sem recorrer a bibliotecas de alto nível (como **requests** ou **urllib**).

Exemplo de Entrada e Saída:

- Entrada:

  URL: `https://www.exemplo.com/pagina/index.html`

- Saída:

  HOST: `www.exemplo.com`

  Caminho: `/pagina/index.html`

  Nome do Arquivo: `index.html`

  HEADER: O Header da resposta deverá ser salvo em `header_resposta.txt`

  CONTENT: Salvo no arquivo `index.html` (ou outro formato dependendo do tipo de conteúdo).