Relatório do Trabalho de Redes de Computadores

Douglas Leite de Souza, Talles Takeshi Takagi e Yuri Karan Benevides Tomas

October 24, 2014

Abstract

Este trabalho tem por objetivo a implementação da funcionalidade básica de um servidor Web. O servidor deve permitir que clientes HTTP (Firefox, IE, etc.) se conectem ao servidor e façam downloads de arquivos. O servidor para o protocolo HTTP deve implementar os métodos de consulta GET e POST usando os protocolos TCP/IP para transferência das páginas HTML.

1 Implementação

1.0.1 Arquivos do código-fonte

DirectoryManager.h
Contém a classe *DirectoryManager*, responsável pela manipulação de diretórios.

Error.h
Contém a classe *Error* que organiza a impressão de mensagens de erro.

GetExtension.h
Contém a classe *GetExtension* que possui dois métodos estáticos: *getExtension* e *getMIME*. Eles retornam a extensão do arquivo passado como parâmetro e descobrem o MIME(Multipurpose Internet Mail Extensions) à partir da sua extensão, respectivamente.

HTTP.cpp
Possui o cabeçalho de diversas métodos da classe *HTTP* usados na interpretação do request para a criaçao do response. Além disto possui a implementação do método *execute* da classe HTTP.

HTTP.h
Possui a implementação de diversos métodos da classe *HTTP*.

main.cpp

ParserHTTP.h
Possui a classe que transforma a string recebida no request para um objeto da classe *RequestHeader*.

RequestHeader.h
Contém a classe *RequestHeader* que é que contém o conteúdo de um request tratado e separado entre os seus atributos

ServerLog.h
Responsável pela implementação do log do servidor.

1.0.2 Descrição da implementação

O nosso servidor web foi implementado utilizando a linguagem *C++*. Utilizamos a *IDE Eclipse* para realizarmos a codificação e os programas foram compilados utilizando o *g++*.

Após pegar todo o request através da função recv da biblioteca sys/socket, tratamos o seu conteúdo através da classe *ParserHTTP*, que transforma seus dados em um objeto da classe *RequestHeader*. Após isto, um objeto da classe *HTTP* trata o request retornando o response adequado.

O método execute da classe *HTTP*, dependendo da versão do protocolo utilizado pelo navegador e do método recebido no request, envia a tarefa de criar o response para um função específica.

Caso o método do request não seja *GET* ou *POST*, como retorno da *doGet* enviamos o retorno do método *doBadRequest*. O método *doBadRequest* retorna um response padrão para a resposta 400.

Caso a versão do protocolo *HTTP* não seja a 1.1, o retorno da *doGet* será o retorno do *doVersionNotSupported*. O método *doVersionNotSupported* retorna um response padrão para a resposta 505.

O método *doGet* da classe *HTTP* trata um *GET* verificando a *URI* passada no request. Se ele se referir a uma pasta, o retorno da *doGet* será um response com um html contendo os itens da pasta no *message-body*. Se ele se referir a um arquivo, o campo *message-body* do response recebe o conteúdo do arquivo referente. Caso o arquivo não seja encontrado, o retorno da *doGet* será o retorno da *doNotFound*, que retorna um response padrão para resposta 404.

O método doPost interpreta uma requisição *POST*. Na requisição *POST*, mas especificamente no *message-body* do request, são enviados os campos do formulário e os dados preenchidos pelo usuário. A partir desses dados uma página html é criada virtualmente, ou seja, não há o arquivo físico em si, somente um html enviado na área de dados do response. Para cada campo do formulário são feitas as associações campo, valor. A *URI* da página de resposta é baseada na *URI* contida no paramêtro *ACTION* do *POST*.

2 Funcionalidades especificadas

As seguintes funcionalidades foram especificadas:

2.1 Status-Code do Response

As seguintes respostas deveriam estar implementadas no seu programa:

2.1.1 200 OK

A Requisição foi bem sucedida e a informação foi retornada ao requisitante.

2.1.2 301 Moved Permanently

O objeto requisitado foi movido permanentemente; a nova URL é especificada no campo Location do cabeçalho de resposta da mensagem. O cliente será redirecionado automaticamente para a nova URL.

2.1.3 400 Bad Request

Um código de erro genérico para uma requisição que não é entendida pelo servidor.

2.1.4 404 Not Found

O documento requisitado não existe neste servidor.

2.1.5 505 HTTP Version Not Supported

A versão do protocolo de requisição não é suportada por este servidor.

2.2 8080 como porta padrão

Ao executar o servidor deve-se indicar o número da porta em que ele receberá conexões. Se nenhuma porta for especificada, o servidor deve escutar na porta 8080.

2.3 Método GET

Seu servidor deve ter suporte ao método GET para requisitar sites armazenados no diretório que conterá sites na linguagem HTML. Deve ser implementado um site em HTML: index.html, contendo um pequeno texto e uma foto.

2.4 Método POST

Seu servidor deve ter suporte ao método POST para enviar dados de formulários para páginas WEB. Deve ser implementada uma página de exemplo para testes, chamada de post.html que contenha o formulário. Ao tratar essa página o servidor deve retornar outra página como resposta contendo as informações enviadas no formulário.

2.5 Diretórios

Você deve adicionar ao seu servidor a capacidade de navegação em diretórios. Se o documento requisitado for um diretório, seu servidor HTTP deve retornar um documento HTML com links para os arquivos/diretórios presentes no diretório. Você deve permitir navegação recursiva de diretórios. Estudem as páginas de manual das funções opendir e readdir.

2.6 Concorrência

O servidor deve ser capaz de atender vários clientes simultaneamente. Assim deve ser implementado um modelo de concorrência, seja usando processos ou threads. Todos os tratamentos devem ser realizados de forma a não deixar filhos zumbis, ou threads paradas.

2.7 Arquivo de Log

Seu servidor deve manter um registro das requisições atendidas, indicando o IP do cliente solicitante, o método e objeto requisitado e a data e hora.

3 Funcionalidades implementadas

- Status-Code do Response com exceção da resposta 301
- 8080 como porta padrão
- Método GET
- Método POST
- Diretórios
- Concorrência
- Arquivo de Log

4 Funcionalidades não implementadas

- Status-Code da resposta 301

5 Funcionalidades não especificadas extras implementadas

5.1 Diversos formatos de arquivo

O servidor consegue abrir diversos formatos de arquivo como pdf, mp3 e mp4.