Servidor HTTP em C

Eduardo Veiga, Thiago Martins

[1]Universidade Federal do Pampa (Unipampa)
– Bagé – RS – Brasil

{ekv2000,thiagoximendesmartins}@gmail.com

Abstract. *This article it is about a implementation of a http-server in C language as a way of practice theoretical concepts seen in class. Here are details of the implementation of the algorithm, describe how was done the development and we approach ours difficulties and items not who are not implemented*

Resumo. *Este artigo trata da uma descrição da implementação de um servidor HTTP na linguagem C feita como forma de colocar em prática conceitos vistos teoricamente em aula. Aqui apresentamos detalhes da implementação do algoritmo, descrevemos como foi feito o desenvolvimento e abordamos as dificuldades e itens o qual não chegaram a serem implementados*

1. introdução

Um servidor é um sistema de computação que fornece serviços a uma rede de computadores. servidor HTTP é um aplicativo responsável por servir requisições que sigam o protocolo HTTP feita por algum cliente. É um programa que roda em uma maquina e fica atendendo requisições de objetos(arquivos) que são feitas por qualquer usuário conectado ao servidor.

2. Òbjetivos

Afim de exercitar os conceitos vistos em aula, foi proposto a implementação de um servidor HTTP na linguagem C. O servidor deveria suportar multithreading, isto é , servir vários clientes simultaneamente. Enviar dados com taxas máximas definidas, e deveria conter uma cache parametrizável para agilizar o acesso aos dados. E deveria suportar conexões persistentes,isto é, depois de efetuar a transferência de algum objeto qualquer, o servidor não encerra a conexão, pois prevê que o cliente pode requisitar mais objetos

3. Metodologia

Para o desenvolvimento do Servidor inicialmente, dados nossos conhecimentos limitados, realizou-se um estudo em cima da utilização mais aprofundada de sockets em linguagem C.Tambem foi necessário tomar conhecimento de noções de programação com Threads em C, pois nosso conhecimento até então era nulo. Após os estudos iniciais começou-se a de fato esquematizar o projeto do servidor. Cada um iniciou o desenvolvimento individualmente porém a estrutura dos dois trabalhos era similar. Em primeiro lugar era necessário encontrar métodos de se obter uma requisição HTTP, sendo nada mais do que receber uma String de um telnet ou de um browser. A criação de uma estrutura que envia e recebe strings não foi difícil.

A Etapa seguinte foi a de analisar a string que representa a requisição HTTP e extrair dela os dados relevantes. Nesta etapa lidamos apenas com manipulação de Strings, sendo algo de pouca dificuldade pelo fato de não requisitar de conhecimento nenhum sobre sockets.

Com os dados prontos entravamos na parte mais complexa e extensa do servidor, a execução da operação pelo servidor isto é, analisar a requisição e trata-la, procurando o arquivo solitcitado e envia-lo ao usuário se o mesmo existir.

No fim Conseguimos implementar com sucesso o método GET persistente com taxa de envio controlada,porém, infelizmente, não fomos capaz de implementar as threads, por falta de tempo uma vez que a primeira etapa nos tomou muito do tempo que tínhamos.

4. Resultados

O servidor serve adequadamente um usuário por vez em sequencia, quando lhe é solicitado um dos seguintes tipos de arquivos: documentos pdf, imagens jpg textos txt, imagens gif, imagens png e arquivos html. Existe uma quantidade enorme de outros formatos de arquivo o qual o servidor simplesmente não trata, porém tentamos inserir os mais comuns.

O cliente é responsavel pela requisição de um arquivo esta requisição pode vir de um terminallnet ou de um Navegador WEB como o firefox ou ou o Google Chrome

Ex: Telnet, utilizando o padrão a seguir:
Telnet 192.168.0.1 8000
GET /home/usuario/docs/redes.pdf HTTP/1.1

Ex: Browser, utilizando uma URL válida de um arquivo, na central de arquivos do Servidor.
Onde o Browser é responsável por enviar requisições diretamente ao Servidor HTTP.
192.168.0.1:8000/home/usuario/docs/redes.pdf

O servidor não funciona com threads logo ele somente pode atender um cliente por vez em uma conexão persistente, isto é, a conexão se mantém aberta caso o cliente queira requisitar mais objetos do servidor.

O servidor suporta controle da taxa de envio com a utilização de uma variável global que irá delimitar uma taxa máxima para o envio de informação

$define TAXA 1 // taxa de 1MB$
$define TAM_MSG 1025 // numero de bytes servidos$

Para o criação e posterior gerenciamento de sockets, utilizamos uma biblioteca chamada socket.h que contém todas as funções necessárias para a manipulação dos sockets junto a ela utilizamos a biblioteca inet.h que é responsável pela conexão em si.

O servidor implementa corretamente o método GET do http mas não implementa outros métodos como HEAD e POST

5. validação

O servidor foi validado em ambiente linux (gentoo e ubuntu) utilizando os navegadores firefox e google Chrome. Em todos os casos o servidor funcionou corretamente entregando os objetos corretos ao cliente solicitante.

Segue abaixo os passos para compilação e execução do servidor

No terminal digite:
gcc -o servidor servidorHTTP.c
./server 8000 100

8000 = porta de acesso ao socket do servidor.
100 = provável taxa máxima de vazão do servidor.

6. Considerações Finais

Mediante o trabalho proposto, que não é de grande dificuldade, acreditamos que ficamos devendo, pois é de nossa responsabilidade saber gerenciar melhor nosso tempo a fim de que tudo que nos é solicitado possa ser de alguma forma bem atendido. O fato de não termos implementado todas as especificaçoes conclui que a falta de organização no inicio do trabalho acabou afetando o resultado final. Porém podemos compreender na prática detalhes que na aula teórica são de dificil compreensão logo mesmo com recursos faltantes muito foi aprendido.

References

F. Kurose,K. W. Ross(2003). Redes de Computadores e a Internet Uma Abordardagem Top-down Addison Wesley Edito, 3ed

A. S. Tanenbaum(2003). Redes de Computadores Campus Editora, 4ed

SG.Gracioli (2011) http://www-usr.inf.ufsm.br/ giovani/sockets.html