Universidade Federal
de São João del-Rei

TRABALHO PRÁTICO 2

Lucas Gonçalves Nojiri
Arthur Antunes Santos Silva

Neste trabalho iremos implementar um servidor web, baseado no código em C para a disciplina Redes do curso de Ciência da Computação da Universidade Federal de São João del Rei.

São João del Rei

Outubro de 2022

Sumário

1 Introdução

Neste trabalho vamos implementar um servidor web, baseado no código em C apresentado em aula, utilizando 4 técnicas distintas de programação com sockets, dentre estas quatro técnicas o servidor deverá ser iterativo, com 1 socket abrindo por vez e será fechado para a próxima conexão, utilizando threads ou fork e com fila de tarefas, que ao abrir a conexão, o processo principal infileira o socket em uma fila de tarefas e um servidor concorrente.

1.1 Requisitos

As implementações devem ser feitas na linguagem C (C++ pode ser usado para o tratamento de strings), usando a biblioteca padrão da linguagem. Os servidores web serão testados utilizando o software siege.

1.1.1 SERVIDOR ITERATIVO

Neste servidor apenas um socket deve ser aberto por vez, quando este processamento da conexão for terminado, este socket é fechado e a próxima conexão poderá ser aceita.

1.1.2 SERVIDOR FORK OU THREAD

Quando for aceito uma conexão, um processo ou a thread filho quando é criado, é responsável por responder a conexão.

1.1.3 SERVIDOR THREAD E FILA DE TAREFAS

Após a conexão ser aceita o processo principal realiza o enfileiramento dos sockets em uma fila de tarefas. Um número fixo de threads vai ser responsável por responder as requisições infileiradas, com o modelo produtor/consuimdor.

1.1.4 SERVIDOR CONCORRENTE

Esteservidor utiliza o "select"para esperar simultaneamente em todos os sockets que estão abertos e utiliza o processo somente quando novos dados estão chegando.

2 Testes

Foi feito um planejamento para os testes dos algoritmos, incialmente foi definido que seriam feitos testes com número de usuários e tempos diferentes usando o software Siege. Porém foram encotrados alguns erros na execução do sofware que não conseguimos corrigir.

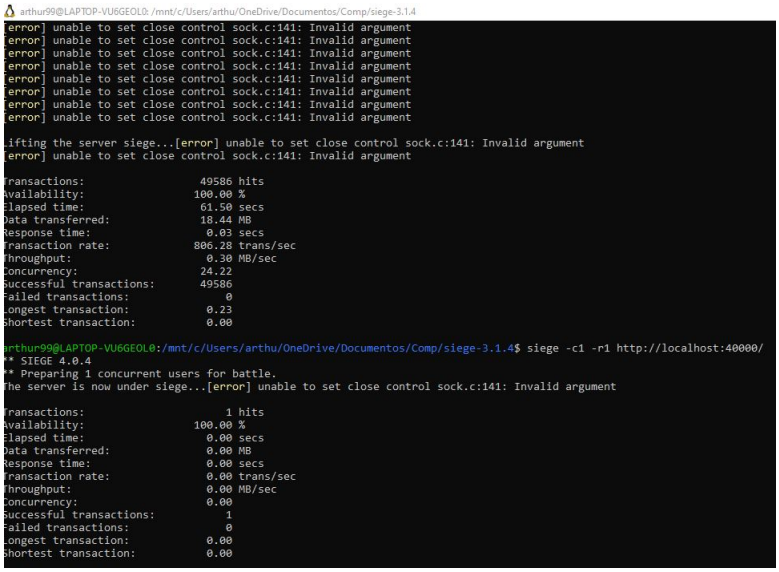

Figura 1: Testes no software Siege

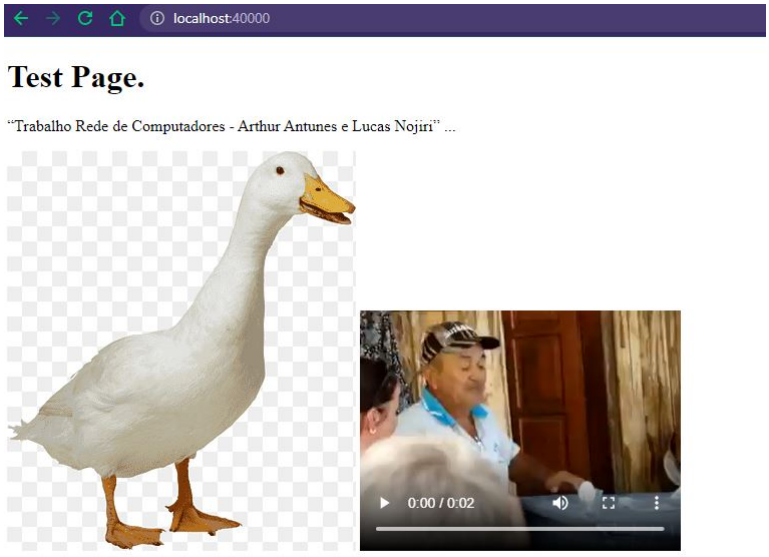

Figura 2: Teste servidor

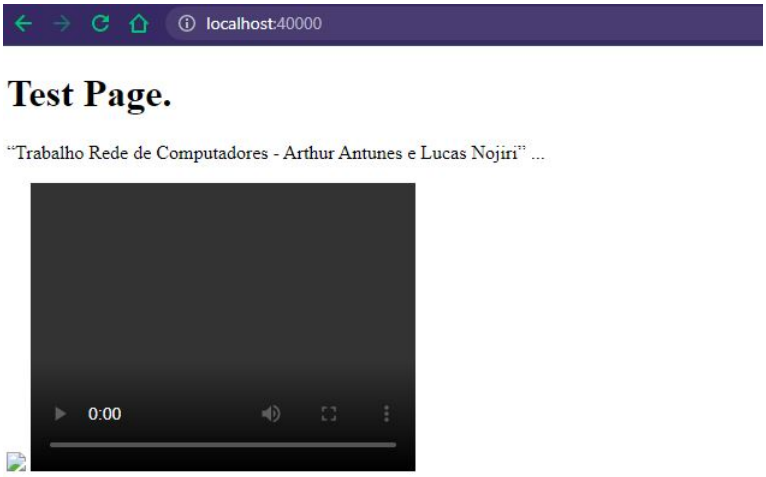

Figura 3: Página não carregava completamente

Porém ainda sim foi possível fazer as conexões nos servidores criados. A seguir estarão detalhados o desempenho de cada técnica implementada.

- Servidor Iterativo: foi funcional todas as vezes em que o cliente se conectou.

- Servidor utilizando fork ou thread: foi funcional maioria das vezes em que o cliente se conectou, porém apresentava o erro(figura 3).

- Servidor utilizando threads e fila de tarefas: foi extremamente instável, e o cliente não se conetava. Era visivelmente o mais lento. Apresentou o erro 404 e o da Figura 3.

- Servidor concorrente: foi extremamente instável, assim como o servidor acima, e o cliente não se conetava em todos os testes.

3 Descrição dos algoritmos e estruturas de dados

3.1 Servidor

Esta seção é responsável por armazenar as solicitações de conexão e configurações da estrutura do servidor. Nele são administradas as informações sobre os arquivos dos sockets criados, e com os outros métodos implementados com fork e threads.

3.1.1 criaSocket (int qtde$_con$)

Este módulo é responsável pela estrutura armazena e realiza a abertura do socket para ouvir as solicitações de conexão. Além disto é verificado se o socket ja está em uso e realiza a configuração da estrutura do servidor, atribuindo a familia de protocolos da internet, recebe conexões de qualquer endereço e seta a porta para rodar o processo, após isto cria um link entre a estrutura servidor ao ID do socket. E por fim é feita uma limitação do número de conexões a uma conexão por vez.

3.1.2 Requisicao *readHeader (int clienteSocket)

Realiza a leitura do cabeçalho do socket cliente.

```
1    Requisicao *req = malloc (sizeof(Requisicao));
2    req->bytes_lidos = 0;
3    req->bytes_lidos = read (socket_cliente, req->cabecalho, HEADER_SIZE);
4    return req;
```

3.1.3 Resposta *carregaArq (char *url)

Este módulo é responsável para carregar o arquivo, se o arquivo não for encontrado, ele será categorizado como "Not found"na leitura do arquivo 404, recebendo o tamanho do arquivo e sendo carregado para a memória, caso ele exista será feito a leitura do arquivo solicitado, junto com os dados a função recebe o tamanho do arquivo.

Caso não haja memória suficiente para o arquivo, o file descriptor será fechado, e é realizado a leitura do arquivo 413, que tambem tem o tamanho do arquivo.

3.1.4 Resposta *httpHeader (Requisicao *req)

Este módulo é responsável por receber a estrutura do arquivo e realiza a definição das estruturas da mensagem, copiando o url do cabeçalho e tenta realizar a abertura do arquivo. Se o arquivo que foi solicitado for o index ou um diretório/arquivo não regular, o diretório será fechado.

```
1    if ((url[0] == '/' && url[1] == 0) || url[0] == 0 || d != NULL){
2    closedir (d);
3    arquivo = carregar_arquivo ("www/index.html");
```

3.1.5 void clienteResp (int clienteSocket))

Este módulo é responsável por receber o cabeçalho do cliente de requisição e declara as estruturas, depois as estruturas de conexão saõ liberados.

3.1.6 void clienteRespThread (void *args)

Esta função, responde os sockets dos clientes que são requisitados.

3.1.7 void iterativo (void))

Este módulo é responsável por manter o servidor ativo, realiza a conexão do cliente e o responde, logo após isto o socket que escuta o cliente é encerrado.

3.1.8 void paralelo (void)

Neste módulo o socket escuta as conexões, inicia conexão com o cliente, junto as estruturas de threads, se o sistema sobrecarregar as threads devem ser terminadas. E o servidor escutado no socket é encerrado.

3.1.9 void consumidor (void) e void produtor (void)

```
1     int socket_escuta, socket_cliente;
2     int sizeSockaddr = sizeof(struct sockaddr_in);
3     struct sockaddr_in cliente;
4     socket_escuta = criar_socket_escuta (QTDE_CONEXOES);
```

Desta forma ele inicia a fila de clientes com a estrutura de threads, (sendo os consumidores) e realiza a conexão com o cliente, após percorrer o buffer e já estiver atendida, o socket da estrutura do cliente será encerrado e as estruturas são liberadas. Enquanto este servidor estiver ativo as requisições na fila de conexões serão atendidas e marcadas, porém se nenhuma requisição fr atendida, haverá um pause, e ele vai dormir por 0.2 micro-segundos

3.1.10 void concorrente (void))

```
1     int socket_escuta, socket_cliente;
2     int max_socket, i, atividade;
3     int size_sockaddr = sizeof (struct sockaddr_in);
4     struct sockaddr_in cliente;
5     fd_set read_fds, master;
6     FD_ZERO (&master);
7     FD_ZERO (&read_fds);
```

O socket será escutado pelo set master, atualizando o valor máximo do conjunto de scokets como o socket do accept. Enquanto o servidor estiver ativo, ele vai esperar por algum novo cliente. Se algum socket foi escrito, ele vai escutar e receber novos clientes e se a conexão for aceita sem erros ele vai responder.

3.2 Server.h

Este módulo é responsável por armazenar as bibliotecas utilizadas para a implementação.

1. **int criaSocket (int qtde_con);** Este módulo é responsável pela estrutura armazena e realiza a abertura do socket para ouvir as solicitações de conexão.
2. **Requisicao *readHeader (int clienteSocket);** Realiza a leitura do cabeçalho do socket cliente.
3. **Resposta *carregaArq (char *url);** Este módulo é responsável para carregar o arquivo.
4. **Resposta *httpHeader (Requisicao *req);** Este módulo é responsável por receber a estrutura do arquivo e realiza a definição das estruturas da mensagem.
5. **void iterativo (void);** Este módulo é responsável por manter o servidor ativo, realiza a conexão do cliente e o responde
6. **void clienteResp (int clienteSocket);** Este módulo é responsável por receber o cabeçalho do cliente de requisição e declara as estruturas
7. **void clienteRespThread (void *args);** Esta função, responde os sockets dos clientes que são requisitados.
8. **void paralelo (void);** Neste módulo o socket escuta as conexões, inicia conexão com o cliente, junto as estruturas de threads
9. **void concorrente (void);** O socket será escutado pelo set master, atualizando o valor máximo do conjunto de scokets como o socket do accept.
10. **void produtor (void); / void consumidor (void);** Enquanto este servidor estiver ativo as requisições na fila de conexões serão atendidas e marcadas, porém se nenhuma requisição fr atendida, haverá um pause, e ele vai dormir por 0.2 microsegundos

3.2.1 Estruturas Pthread

```
1 pthread_t threads[QTDE_CONEXOES];
2 u_int32_t qtde_requisicoes;
3 pthread_mutex_t qtde_requisicoes_protect;
4 pthread_mutex_t requisicoes_protect[BUFFER_SIZE];
```

3.2.2 Struct Requisição

Estrutura de requisição.

```
typedef struct {
    int bytes_lidos;
    char cabecalho[HEADER_SIZE];
} Requisicao;
```

3.2.3 Struct Resposta

Estrutura de resposta do buffer.

```
typedef struct{
    u_int32_t tamMsg;
    u_int8_t status;
    char *bufferResp;
} Resposta;
```

3.2.4 Struct clientent

Estrutura de retorno quando se aceita uma conexão com usuários.

```
typedef struct{
    int sockfd;
    struct sockaddr_in sock_addr;
    socklen_t socklen;
}clientent;
```

4 Conclusão

Neste trabalho, aprendemos como é o funcionamento de um servidor web com a implementação em C com a utilização das 4 técnicas utilizadas, os conceitos sobre os processamentos de conexão, os processos "filho"para responderem a conexão, a utilização de threads e forks, a utilização de sockets em filas de tarefas e seus procedimentos para o enfileiramento das tarefas

Este trabalho foi de grande importância para a visualização mais prática de todo o conteúdo que havíamos aprendido em aula. Fazendo com que o buscássemos formas de utilizar estes conhecimentos, além de trazer um desafio na áera de programação, também nos trouxe muito conhecimento sobre a parte da implementação e como desenvolver um servidor web em C.

5 Referências

https://github.com/warSantos/REDES02

http://www.joedog.org/index/siege-home

http://ziutek.github.com/web

https://www.codigofonte.com.br/codigos/web-server-em-c

https://www.vivaolinux.com.br/topico/C-C++/Como-usar-HTML-com-linguagem-C

https://acervolima.com/analisador-de-html-em-c-c/