RELATÓRIO DO PROJETO

Sistemas Operativos – Engenharia Informática

Servidor HTTP

• U  C •

Realizado por:

Miguel Machado, 2014214547 | Teresa Salazar 2015249122

INTRODUÇÃO

O objetivo deste trabalho é implementar um servidor web na linguagem C, capaz de servir vários pedidos em simultâneo, quer a páginas estáticas, quer a páginas comprimidas. Também é necessária a existência de uma consola de configurações e gestão de estatísticas.

CONFIGURAÇÕES INICIAIS E TERMINAÇÃO DO SERVIDOR

O processo principal começa por criar a memória partilhada através da função *create_shared_memory()*. Posteriormente, é criado o processo das estatísticas através da função *create_processes()*. Depois, é feito o 'attach' das estruturas 'config' e 'last_request' com *attach_shared_memory()*.

Com a função *configuration_start()*, é lido o ficheiro config.txt e atualizada a estrutura config com o porto para o servidor, a política de escalonamento, o número de threads a utilizar para tratamento de pedidos pelo servidor e a lista de ficheiros comprimidos autorizados (definidos pelo nome do ficheiro).

Depois, é criado a estrutura do buffer de pedidos efetuados pelo cliente (lista ligada) com *create_buffer()*, que cria uma lista ligada. Cada Request (pedido) é uma struct constituída por: type, conn (que representa a conexão com o cliente), required_file, get_request_time, serve_request_time e ponteiro para o nó seguinte.

Posteriormente, são criados os semáforos, a thread do pipe e a pool de threads. Desta forma, as threads estão prontas a servir pedidos do cliente.

Depois é declarado um sinal do tipo SIGINT para o servidor estar preparado para terminar, após ser feito Ctrl-C. Nessa altura, o servidor começa por fechar as sockets, espera que todos os pedidos terminem, termina os processos, e fazer a limpeza no sistema de todos os recursos partilhados, apaga o buffer e os semáforos.

RECEÇÃO DE PEDIDOS

O processo principal é responsável por receber as ligações HTTP e escalonar o tratamento dos pedidos efetuados pelos clientes. Através da função *accept()* é aceitada uma conexão na socket.

Depois de identificado o cliente pelo address e o porto, o request do cliente é processado.

SCHEDULER E TRATAMENTO DE PEDIDOS

Através da função *page_or_script()* é verificado o tipo de conteúdo do request: estático ou comprimido. Com a função *compressed_file_is_allowed(*) é verificado se, caso o conteúdo seja comprimido, ele está presente na lista de ficheiros comprimidos autorizados no ficheiro config.txt. Caso o ficheiro seja comprimido e autorizado ou o ficheiro seja estático, verifica-se se há espaço no buffer e se há alguma thread disponível. Se sim, o pedido é adicionado ao buffer e é enviada uma resposta ao cliente na forma de uma resposta HTTP através da respetiva socket. No entanto, caso não haja capacidade para servir o novo pedido, é apresentada uma mensagem de erro ao cliente.

Se se trata de um acesso a conteúdo estático (página HTML) o ficheiro em causa é lido, e o seu conteúdo devolvido ao cliente. Se se tratar de um acesso a conteúdo estático comprimido, o ficheiro é descomprimido através da função *execute_script()*, que envia um comando para a shell que descomprime

o ficheiro e apaga o comprimido. Assim, será possível enviar ao cliente o resultado da execução também através de uma resposta HTTP.

Os pedidos são inseridos de forma ordenada conforme o tipo de scheduling. De acordo com a política definida na configuração do servidor, o scheduler vai controlar a forma de como os pedidos são servidos. Na altura de retirar o pedido do buffer por uma thread, é só necessário retirar o request da última posição do buffer.

CONSOLA DE CONFIGURAÇÕES

Existe uma console de configurações que é utilizada para alterar a forma de como servidor atua de forma dinâmica. Esta consola, presente no ficheiro console.c, tem o seu próprio executável que comunica com o processo principal através de um named pipe. O processo principal está sempre pronto para receber instruções do named pipe.

Os comandos do utilizador são lidos pela consola e escritos no named pipe. Primeiro é pedido ao utilizador que escolha a opção do menu que pretende. Se este escolher a opção "1", poderá alterar o tipo de scheduling. Aí, o scheduler vai ordenar o buffer de acordo com a política pretendida, utilizando para tal um bubblesort. Caso escolha a opção "2", pode alterar o número de threads. Se o novo número de threads for maior, criam-se novas threads. Caso contrário eliminam-se um certo número de threads. Se escolher a opção "3", pode alterar a lista de ficheiros comprimidos autorizados.

É de salientar que as funções de leitura dos comandos do utilizador estão devidamente protegidas para que o programa não deixe de funcionar caso o utilizador não introduza um input válido.

INFORMAÇÃO ESTATÍSTICA

No ficheiro mmf.log, estão armazenadas informações relativas ao funcionamento do servidor. Em cada linha há informação sobre: tipo de pedido (static ou compressed), ficheiro HTML lido, tempo em milisegundos de receção do pedido e tempo em milisegundos de terminação de servir o pedido (após envio do resultado ao cliente).

Como temos um processo para as estatíticas, este está em loop a verificar se há um novo pedido para ser escrito. Quando há, é enviada informação sobre esse pedido e escrita no ficheiro mmf.log.

Sempre que é enviado um sinal do tipo SIGUSR1, é chamada a função *print_statistics()*, que imprime no ecrã o número total de pedidos servidos a páginas estáticas e a ficheiros comprimidos e o tempo médio para servir um pedido a conteúdo estático não comprimido e a conteúdo estático comprimido.

Após a receção de um sinal do tipo SIGUSR2, o processo de gestão de estatísticas faz um reset à informação estatística agregada, eliminando toda a informação do ficheiro mmf.log.

CONCLUSÃO

Com este trabalho aprendemos o funcionamento geral de um servidor HTTP, explorando diversos mecanismos de gestão de processos e memória, bem como a sua sincronização utilizando semáforos e mutexes.

Pensamos que o projeto está bem conseguido e que cumprimos todas as componentes de avaliação necessárias.