UNIVERSIDADE DE COIMBRA

Servidor HTTP

Sistemas Operativos

Licenciatura em Engenharia Informática

2014/2015

Autores:

João Paulo dos Reis Gonçalves	2012143747
João Miguel Borges Subtil	2012151975

Arquitectura

Foi implementado um servidor HTTP com suporte de multi-threading, acesso a informação estática e dinâmica, gestão de configurações e de estatísticas.

Estruturas

Nesta implementação foram utilizadas 4 estruturas.

```
typedef struct configuration{
    int server_port;
    int n_threads;
    int scheduling_policy;
    char scripts_list[5][10];
}Configuration;
```

```
typedef struct thread_pool
{
        pthread_t mythread;
        sem_t *sem;
        int occupied;
        int position;
}Thread_pool;
```

```
typedef struct request
{
    int type;
    char file[SIZE_BUF];
    int client_socket;
    int dispatched;
    int occupied;
}Request;
```

```
typedef struct message
{
        long mtype;
        int type;
        char file[SIZE_BUF];
        int n_thread;
        char reception_hour[20];
        char final_hour[20];
}Message;
```

A primeira contém os parâmetros de configuração do servidor, a segunda representa cada elemento da pool de threads, a terceira cada pedido a ser atendido e a quarta estrutura é utilizada como mensagem a enviar do processo principal ao processo de gestão de estatísticas pela message queue.

Descrição do funcionamento do programa

Processo principal

O processo principal começa por criar e inicializar os recursos necessários à execução do servidor (semáforos, memória partilhada, processo de gestão de configurações e processo de gestão de estatísticas, ficheiro de logs, message queue, a thread de escalonamento e a pool de threads). Além disso, é configurado o "listening port" e é feita a prevenção de interrupções indesejadas por sinais.

De seguida, o processo principal fica a aceitar pedidos e caso haja espaço no buffer de pedidos (array de estruturas Request) estes são lá colocados. Se o buffer de pedidos estiver cheio, o cliente é informado através da função buffer_full e a ligação com o servidor termina. Por cada pedido novo aceite, o processo principal alerta a scheduler_thread.

Esta vai determinar qual é o próximo pedido a ser despachado, consoante a actual política de escalonamento definida na configuração do servidor e conforme um pedido seja válido ou não, ou seja, o pedido tem de ser de um ficheiro existente ou de um script em que a sua execução está autorizada. Esta verificação é feita através da função is_valid. Caso o pedido não seja válido ou haja algum erro a executá-lo, o cliente é informado disso através da função not_found, not_authorized ou cannot_execute e a ligação entre o cliente e o servidor termina. Além disto, a scheduler_thread determina qual é a thread que vai despachar o pedido.

Este mecanismo da scheduler_thread só executar quando for alertada de que há novos pedidos é feito através do semáforo stop_scheduler.

As threads que despacham pedidos fazem parte de um array de estruturas do tipo thread_pool e a sua

“start routine” é a função dispatch_requests. Cada estrutura contém uma thread, um semáforo, um inteiro occupied para indicar se esta thread está em execução ou à espera de pedidos e outro inteiro position que indica a posição no buffer de pedidos do pedido que vai ser atendido pela thread. Quando estas são criadas ficam à espera de serem alertadas pela scheduler_thread para despacharem um pedido.

O semáforo threadlist é inicializado com o número de threads que a pool contém. Quando todas as threads estão ocupadas, este bloqueia a scheduler_thread para evitar esperas activas. Caso o valor deste semáforo ainda não seja 0 após fazer sem_wait, a scheduler_thread percorre a pool de threads e a primeira encontrada com occupied igual a 0 faz sem_post no semáforo associado a essa thread.

Caso o pedido seja executado com sucesso pela thread, esta envia através de uma message queue uma mensagem para o processo de gestão de estatísticas com a informação necessária a guardar no ficheiro de logs (estrutura Message).

Quanto à execução de Shell scripts, a thread encarregue de atender o pedido cria um processo filho que vai executar o script e enviar o resultado dessa execução de volta para a thread, que de seguida envia para o cliente. Esta comunicação entre a thread e o processo filho é feita através de um pipe. No processo filho o pipe duplica o descritor de escrita na posição do stdout enquanto a thread lê do pipe.

Processo de gestão de configurações

O processo de gestão de configurações começa por ler as configurações do ficheiro “serverconfigs.txt” e colocá-las na memória partilhada. Para garantir que o processo de configurações acaba de ler as configurações antes de a execução voltar para o processo principal é usado um semáforo chamado loadfileschecker. De seguida é feita a prevenção de interrupções indesejadas por sinais e por fim este processo fica à espera de receber um sinal do tipo SIGHUP.

Quando recebe um sinal desse tipo chama a função catch_sighup_config que permite alterar as configurações do servidor com o mesmo em funcionamento e guardar as novas alterações no ficheiro “serverconfigs.txt”. Após a alteração (ou não) das configurações, é enviado ao processo principal um sinal do tipo SIGUSR1 para que este reinicie o servidor através da função restart_server.

Processo de gestão de estatísticas

O processo de gestão de estatísticas começa por prevenir interrupções indesejadas através de sinais e de seguida fica à espera de receber mensagens que vão ser guardadas no ficheiro de logs. Este verifica o tipo do pedido associado à mensagem e actualiza o número de pedidos atendidos desse tipo, sendo de seguida efectuada a escrita no ficheiro de logs. Para contabilizar o número de pedidos recusados, quando estes são recusados, é enviada uma mensagem “dummy” apenas com o parâmetro type indicando que é do tipo rejeitado, do processo principal para o processo de gestão de estatísticas.

Quando este processo recebe um sinal do tipo SIGHUP, este mostra as estatísticas do servidor (hora de arranque do servidor e hora actual, número total de acessos a conteúdo estático e dinâmico e número total

de pedidos recusados) e o conteúdo do ficheiro de logs.

Terminação do servidor

Quando o processo principal recebe um sinal do tipo SIGINT, ou qualquer um dos três processos recebe algum sinal crítico à execução do servidor, este chama a função catch_signals, que informa que o servidor vai terminar chamando a função clean_up que faz a limpeza de todos os recursos usados pelo servidor e termina os processos.

Estrutura dos ficheiros

O ficheiro de logs (logs.txt) tem em cada linha informação sobre um acesso HTTP com os seguintes campos separados por vírgulas e pela seguinte ordem : tipo de pedido, ficheiro HTML lido ou script executado, número da thread na pool responsável por atender o pedido, hora de recepção do pedido pela thread, hora de finalização após ter enviado o resultado ao cliente.
O ficheiro de configurações (serverconfigs.txt) tem na primeira linha separado por ponto e vírgula e pela seguinte ordem: porto para o servidor, número de threads da pool, política de escalonamento. Nas restantes linhas tem os nomes dos scripts autorizados a executar, um por linha.

Como usar o programa

Para criar os executáveis basta executar o makefile. Para correr o servidor faz-se num terminal "./simplehttpd". Para fazer pedidos de acesso a conteúdo estático deve-se escrever na barra de endereços: "localhost:<porto_associado ao_servidor>/<nome_ficheiro>". Para fazer pedidos de acesso a conteúdo dinâmico deve-se escrever na barra de endereços: "localhost:<porto_associado ao_servidor>/cgi-bin/<nome_ficheiro>".