Projeto SO – Servidor DNS

O projeto de Sistemas Operativos consiste na construção de um servidor DNS(Domain Name System), tendo como base um código que estabelece uma ligação entre o servidor e um ou vários clientes e permite o envio de pedidos por parte de um cliente e a sua receção por parte de um servidor que, posteriormente, envia uma resposta de volta ao cliente. Para a resolução deste problema, utilizamos os conhecimentos adquiridos ao longo do semestre sobre mapeamento de memória, processos e comunicação entre eles, sinais, sincronização e threads.

Estruturas:

Foram criadas várias estruturas para auxiliar a resolução do problema, entre elas temos uma "Data", um "Info_Pipe" e um "Server_info":

- A "Data" armazena vários inteiros e é utilizada para guardar a data e hora de arranque do servidor e da última informação obtida pelo mesmo;
- A estrutura "Info_Pipe" guarda os dados que vão ser transferidos pelo pipe e que liga o processo que gere pedidos ao processo que gere e imprime no ecrã as configurações. Apesar da escrita e leitura do pipe serem sempre sobre a forma de String, esta estrutura facilitou o manuseamento de dados;
- O nosso "Server_info" é usado para ler a informação escrita no ficheiro "config.txt" e guarda cada um dos parametros do ficheiro, numa variável. Com a necessidade de gerir o modo de manutenção, críamos um "array de booleans" que vai informar se a manutenção foi iniciada ("flag[0]=TRUE"), em casa afimativo se já foi cancelada ("flag[1]=TRUE"). O último índice deste array tem a função de informar se o programa já começou para que, caso o programa entre em manutenção, o ficheiro "config.txt" seja lido de novo sem alterar o número de threads e o Named Pipe para onde vamos enviar as estatisticas. Apesar dos componentes do ficheiro de configurações não estarem relacionados com a flag, encontram-se na mesma estrutura para facilitar o mapeamento dos mesmos em memória;
- Criámos também a "Lpedido_no", um nó que vai guardar alem do pedido, as variaveis necessárias à resposta do servidor ao cliente, sendo essa resposta o endereço IP, e contém um ponteiro ("Lpedido"), que aponta para um outro nó, criando assim uma lista.

Funções e ficheiros:

O ficheiro “main.c” contém a função main() que inicializa as estruturas, cria e mapeia a memória, cria semáforos e é resposável pela criação de processos (cada um responsável por um dos gestores).

O “auxiliar.h” contém as estruturas do nosso programa, assim como as variáveis globais.

O “gestor_estatisticas.c” contém a função gestor_estatisticas() que se limita a ler constantemente o pipe. No entanto o pipe é unidirecional e só lê quando um outro processo escrever nele, por isso usamos a função func() que é ativada de trinta em trinta segundos por um sinal de alarme e escreve no pipe. Só quando esta função escreve no pipe é que o gestor de estatísticas escreve no ecrã.

No ficheiro “gestor_configuracao.c” abrimos o ficheiro “config.txt” e lemos os dados do mesmo. Consoante a flag, tal como foi explicado na estrutura “Server_info”, se o valor do índice 2 do array(flag[2]) for “True”, então o servidor ainda não leu o ficheiro e, por isso, ao ler vai utilizar o valor das threads. Caso o valor seja “FALSE”, então o número de threads e o Named Pipe não podem ser alterados, sendo os valores associadas a variáveis, para facilita a leitura (sequencial), e são descartadas.

No map_file.c está a funçao(“map_localdns()”) usada para mapear o ficheiro “localdns.txt” em memória.

O ficheiro “gestor_pedidos.c” foi o mais díficil de organizar, porque tinha de conter as funções que permitiam o tratamento de pedidos. Para esta gestão era preciso não só verificar o tipo de domínio (validade) e condicioná-lo com o modo de manutenção, mas também era necessário obter o endereço IP, por isso foram usadas algumas condições e funções, como o verifica_validade(), que permitem saber se o pedido realizado pelo cliente é do tipo local, verificando se o nome coincide com algum dos domínios presentes no “config.txt” e que estejam relacionados com os domínios locais. Caso esta validade não se verifque, faz-se o mesmo para verificar se há relação com um domínio é externo, não havendo conclui-se que ele é inválido. O uso de “bool flag[3]” foi bastante útil para verificar se o servidor estava em modo de manutenção e, assim sendo, recusar os pedidos não locais. São utilizadas listas para controlar a resolução de pedidos, porque com elas podemos não só separar os domínios locais de externos e, posteriormente, podemos ler as listas e controlar a ordem de resposta aos pedidos, seguindo o modelo “FIFO” (“first in first out”).
“worker()” é uma função que é executada constantemente pelas threads e que vai enviar uma resposta consoante a prioridade referida no enunciado do projeto. Este modelo de leitura de estruturas, juntamente com as condições “if” utilizadas, permite controlar a prioridade de resposta do nosso servidor. Nestas condições é chamada a função elimina_first, que se limita e eliminar o nó da lista correspondente ao pedido resolvido, para que este não seja resolvido novamente.

O tempo despendido na realização deste projeto rondará as 50 horas.

Trabalho realizado por:
Daniel Azevedo nº 2014200607
Nuno Afonso Santos nº 2014226541