Projeto de Sistemas Operativos 2015/2016

O objectivo deste projeto passa por evoluir um código de um cliente e de um servidor *Trivial File Transfer Protocol* pondo em prática os conhecimentos adquiridos na cadeira de Sistemas Operativos. Sendo que estas alterações visam melhorar o desempenho, escalabilidade e funcionalidade desta aplicação.

Uma primeira versão do código cliente (client.c) e do servidor (tftps.c) são fornecidos. O Servidor (*tftps*) quando é invocado atende todos os pedidos de forma sequencial. Esta implementação não é adequada, nomeadamente quando o servidor recebe simultaneamente um elevado número de clientes. Por outro lado, um cliente pode solicitar em paralelo diversos ficheiros diminuindo o impacto da latência da rede e apresentando ao utilizador um menor tempo de resposta.

Por essas razões, pretende-se que o servidor beneficie de uma implementação multi-processo ou multi-tarefa, aproveitando mais eficazmente o hardware da máquina e atendendo a um maior número de pedidos concorrentes.

O projeto de Sistemas Operativos na componente prática é individual e será dividido em quatro fases. As datas de apresentação serão definidas pelo Docente. Os projetos entregues fora dos prazos ou não apresentados presencialmente, não serão considerados para classificação.

FASE1

Na primeira fase deve implementar um cliente e servidor TFTP com suporte multi-processo.

Deve acrescentar à sua implementação a funcionalidade de listar ficheiros do diretório remoto, fazendo as alterações que considerar necessárias tanto no cliente como no servidor. O cliente deve receber o comando por argumento ("*ls nomeDiretório*") e enviá-lo para o servidor. Por fim, o cliente irá receber uma mensagem de volta com a listagem dos ficheiros contidos na diretória do servidor. Caso a diretoria não exista deverá devolver uma mensagem com essa indicação ao cliente.

O código do cliente deve ser alterado de forma que este solicite ao servidor, de forma concorrente, a transferência de todos os ficheiros do diretório indicado (e.g., mget nomeDiretório – *multiple get*). O cliente vai então lançar *N* processo filho para solicitar *N* ficheiros, e depois aguardar que todos os seus filhos terminem. Por fim, antes de terminar, este deve também imprimir o tempo de execução. Cada filho vai efetuar um pedido ao servidor e guardar o ficheiro recebido num diretório corrente. Deve registar os tempos obtidos, na mesma máquina, entre máquinas.

O código do servidor deve ser alterado de forma a que este inicie um processo para atender cada pedido. O processo principal (que recebe os pedidos) deve ignorar o estado de finalização dos seus filhos (ver sinal SIGCHLD). Depois de atender um pedido o processo filho simplesmente termina.

FASE 2

Na segunda fase o objectivo é tornar o cliente e servidor TFTP em aplicações multi-tarefa. O comportamento do cliente e do servidor HTTP será o mesmo que foi implementado na fase 1, mas desta vez usado tarefas POSIX. Os mesmos testes devem ser efectuados (cf. fase 1). O servidor será alterado para lançar uma tarefa para atender cada novo pedido. Use diretivas de pré-processamento, nomeadamente Macros, para definir se a sua solução recorre ao paradigma de programação multi-processo ou multi-tarefa (e.g., *#define OperationMode 0 // 0 is multi-process or 1 is multi-thread*).

FASE 3

Na terceira fase o servidor deve ser alterado de forma a que coloque em execução *N* tarefas para atender pedidos (*worker threads*). Estas tarefas vão funcionar como consumidoras enquanto a tarefa principal (que recebe os pedidos vindos de Internet) funcionará como produtora. O produtor vai colocar os pedidos num *buffer* de tamanho *M.* Os consumidores retiram do *buffer* os pedidos para os atenderem. Trata-se de uma implementação do problema produtor / consumidor que é revisto nas aulas. Os valores *N* e *M* são passados ao servidor por argumento através da linha de comandos. No fim, deve implementar uma tarefa adicional que imprime periodicamente no terminal o estado atual do *buffer*.

FASE4

Na última fase vamos melhorar novamente o desempenho do servidor. Uma das vantagens de ter um servidor multi-tarefa é que todas as tarefas de determinado processo partilham um conjunto de recursos, como por exemplo, variáveis globais. Uma das atividades do servidor TFTP consiste em servir ficheiros que se encontram armazenados no disco. É possível melhorar o desempenho do servidor implementando uma estratégia de *caching*. A ideia passa por manter em memória os ficheiros mais solicitados evitando diversos acessos ao disco. Quando uma tarefa retira um pedido do *buffer* ela verifica se o ficheiro se encontra em *cache*. Se já lá estiver, entrega-o ao cliente sem aceder ao disco. Caso contrário, vai ler o ficheiro ao disco, carrega-o para a *cache* e serve o cliente. Como a *cache* de ficheiros é partilhada por todas as tarefas, os ficheiros mais populares ficarão, com elevada probabilidade, sempre presentes na *cache* contribuindo para um melhor desempenho do servidor TFTP.

A implementação da *cache* usa uma estrutura de dados designada por *hash table* que será apresentada durante as aulas. No arranque do servidor, antes de este lançar os *worker threads*, deve permitir através de argumentos fazer o pré-carregamento de ficheiros existentes na estrutura de dados (*cache*). Os *worker threads* ficam responsáveis por gerir a *cache* de páginas em memória de acordo com o procedimento descrito no parágrafo anterior.

Esclarecimentos sobre o enunciado deverão ser obtidos, junto do(s) docente(s), no decorrer das aulas, durante as horas de atendimento ou através do correio electrónico.