MESTRADO INTEGRADO EM ENGENHARIA BIOMÉDICA

Sistemas Operativos

Trabalho prático[1]

Ano lectivo de 2011-2012 Duração: entrega até ao Natal de 2011

Gestão de Stocks de Medicamentos

Este trabalho consiste na análise e desenvolvimento de uma aplicação informática para gestão (concorrente) do stock de medicamentos. Deverá ser desenvolvido em linguagem C e executar em ambiente Unix. Além das tradicionais operações de consulta e actualização da quantidade de medicamentos existentes em stock, incluem-se também operações de inserção, remoção e listagem de medicamentos segundo determinados critérios. Deve admitir-se que se trata de um conjunto muito grande de medicamentos que, pela sua dimensão, terá de residir em disco e não cabe na totalidade em memória. Este é um aspecto **central**: pretende-se avaliar o desempenho, portanto deverá carregar e testar o seu sistema com 2 milhões, 20 milhões de medicamentos, os necessários para perceber se a sua solução é *escalável*.

A grande dimensão da "base de dados" acarreta desde logo a necessidade de olhar com cuidado para as questões de eficiência e desempenho (espaço ocupado, tempo de execução, espera passiva, etc.). Assim, deverá ter-se cuidado com a estrutura dos dados, recorrendo nomeadamente à indexação de ficheiros, à utilização de pesquisas binárias em ficheiros ordenados, etc. Outro dos requisitos desta aplicação é a capacidade de lidar correctamente com concorrência, uma vez que podem existir vários utilizadores simultâneos. A segurança também não deverá ser esquecida, sendo necessário pensar no controle de acesso aos dados.

Sendo um trabalho de Sistemas Operativos, pretende-se uma solução *de baixo nível*: se se considerar um sistema constituído por várias camadas, onde no topo estão os utilizadores e na base está o hardware, baixo-nível significa que se pretende trabalhar tanto quanto possível próximo da base. Assim, soluções que envolvam a utilização de bases de dados, linguagens de interrogação de alto-nível (e.g. SQL), linguagens orientadas a objectos, etc., estão excluídas à partida por esconderem grande parte do detalhe que se pretende explorar aqui. De acordo com o objectivo primário da unidade curricular de Sistemas Operativos, o de ajudar a perceber como funcionam os sistemas informáticos, vamos "abrir o capôt do automóvel" e tentar perceber a mecânica do software de sistemas. Podemos contar apenas com a cultura dos Sistemas Operativos clássicos — linguagem C, *system calls* do Unix (Linux, Mac OSX...), ficheiros, interacção entre processos, controle de concorrência, etc. Assim, da próxima vez que alguém

[1] Cotação — 30% nota final

disser que o sistema está lento, que está a gastar muito espaço ou que há problemas de privacidade de informação, ter-se-á uma ideia muito mais precisa do que se passa lá dentro e da forma de ultrapassar esses problemas. Não se esqueça de testar a aplicação com um elevado número de medicamentos; a dimensão da "base de dados" é importante!

O enunciado é razoavelmente vago para fomentar alguma criatividade nos grupos de trabalho e permitir que estes se concentrem nas áreas que acham mais interessantes (por oposição à típica situação de *trabalho prático* = *xarope* que se começa a tomar uns dias antes de acabar o prazo de entrega). Pela mesma razão, outro dos aspectos pouco habituais do trabalho é a forma incremental como deverá ser realizado: ao longo de várias semanas, são sugeridos e começados a resolver durante as aulas pequenos "trabalhos de casa" que não são mais do que componentes do trabalho final. Assim, no final das aulas o trabalho deverá estar praticamente concluído sendo apenas necessário integrar as partes, testar com dados da dimensão pretendida e escrever o relatório final.

Conforme referido, pretende-se desenvolver a capacidade de analisar e especificar problemas, e partir daí para a sua solução por via informática, e ainda a capacidade de lidar com algoritmos concorrentes. Para atingir estes objectivos deverá começar por especificar o problema, definindo os tipos de dados (i.e., responder à pergunta "o que é um medicamento" em termos informáticos?), as operações sobre eles e a forma como os dados são armazenados de forma persistente (ou seja, responder a perguntas como "o que é uma *tabela*? Está indexada? Ordenada? Quantas tabelas vou ter?").

Ainda dentro espírito do baixo-nível e de fazer as coisas *à moda antiga* para ver as vantagens e desvantagens que a subida no nível de abstracção pode trazer, vamos recuar algumas décadas e perceber como seriam os sistemas de gestão de stocks dessa altura. Não havia computadores portáteis nem internet, tipicamente existiria apenas um grande sistema em regime de *time-sharing*: um computador central ao qual estariam ligadas dezenas ou centenas de terminais *dumb*[2]. É esse o cenário deste trabalho, se bem que ele seja agora simulado em portáteis com internet. Vai precisar de vários portáteis, um deles a fazer de computador central onde reside a aplicação e os outros a simularem os antigos terminais VT-100. Se não estiver muito inclinado/a para estudar o acesso remoto via telnet/ssh deverá demonstrar a aplicação a correr simultaneamente em várias janelas, por exemplo, colocando (centenas de) consultas ao mesmo tempo que as inserções de medicamentos e actualizações de stock.

```
$ insere_medicamento guronsan atrib1 atrib2 atrib3
$ top_de_vendas 50
```

A aplicação a desenvolver é na realidade um conjunto de programas escritos em C e activado a partir da *shell*. No exemplo anterior mostra-se a introdução de um medicamento na base de dados (supondo que tem 3 atributos além do nome) e a listagem dos 50 medicamentos mais vendidos. Se pretender criar uma pequena interface com o utilizador, limite-se a um script em Bash que aceita argumentos do utilizador e de seguida invoca os programas compilados (`insere_medicamento`, `top_de_vendas`, etc).

Alguns dos trabalhos de casa úteis para a realização deste trabalho incluem:

[2]Isto significa que a camada da interface com o utilizador era muito pouco apelativa quando comparada com as actuais interfaces gráficas; tinha no entanto a grande vantagem de ser "leve" e funcionar com a largura de banda de antigamente.

- Ordenação de vectores
- Representação interna de *tabelas*
- Criação e comunicação de processos
- Leitura e escrita de ficheiros e pipes
- Partilha de memória, exclusão mútua com semáforos
- Sincronização de processos, e.g. leitores e escritores

O trabalho será portanto um conjunto de programas comprovativos do domínio de técnicas de programação sequencial e concorrente, com **muitas** preocupações de eficiência. Uma vez que neste ano lectivo se exploram em mais profundidade as questões de organização dos dados e avaliação de desempenho das operações sobre eles, não será necessário apresentar nem comparar com uma solução baseada na arquitectura cliente-servidor. Tem apenas de perceber as diferenças entre as duas soluções.

Deverá implementar as operações seguintes:

1. Inserção de medicamentos
2. Remoção de medicamentos
3. Actualização do stock
4. Consulta de stock de um determinado medicamento
5. Listagem de medicamentos mais vendidos

Algumas das questões a considerar são:

- Programação sequencial:
    1. ordenação de vectores de inteiros
    2. "tabelas" da base de dados de medicamentos, ordenação de tabelas
    3. nome do medicamento: dimensão fixa versus dimensão variável?
- Programação concorrente:
    1. identificação das regiões críticas e respectivo padrão de acesso: read/write
    2. operações sobre semáforos
    3. operações sobre memória partilhada

Este conjunto de questões presta-se a alguma divisão de tarefas pelos vários elementos do grupo de trabalho. Isso obriga à especificação prévia dos dados e operações sobre eles para que um elemento possa estar a trabalhar, por exemplo, na garantia de exclusão mútua no acesso ao ficheiros de medicamentos, enquanto outro lida com a listagem do top de vendas. Uma vez testada cada operação em separado (para garantir a correcção funcional), deve passar-se ao teste com vários programas em simultâneo. Pretende-se desta forma cobrir também algumas das boas práticas de projecto de software desenvolvido em equipa e estimular a comunicação entre todos os elementos do grupo, ao invés da mera divisão antecipada do trabalho, ficando cada elemento especialista apenas do seu fragmento.