Universidade do Minho

Mestrado Integrado em Engenharia Biomédica

Plataformas de *Software*

Docentes:

A74727 Ana Ramos

A74753 Ana Sousa

A75088 Ana Machado

Docente:

Francisco Moura

Braga

Dezembro 2017

Resumo

O presente trabalho, realizado no âmbito da unidade curricular de plataformas de *software*, tem como principal objetivo o uso da linguagem C para a implementação de um sistema de leilões. Desse modo, o sistema deverá apresenta gestão da concorrência e alerta os licitadores quando o prazo de licitação se encontra próximo do fim.

A construção deste sistema passou pela implementação de programas que, por exemplo, permitem inserir itens a leiloar e efetuar licitações a um determinado leilão.

Todo o sistema foi projetado tendo em vista uma maximização da sua eficiência quer ao nível do tempo de processamento, quer ao nível do controlo da concorrência. Para otimizar o controlo da concorrência, recorreu-se as funções semáforos.

Índice

1. Introdução

O presente trabalho tem como principal objetivo o desenvolvimento de um sistema, recorrendo a linguagem C, que efetua a gestão de leilões online. Alguns dos aspetos principais a abordar são os seguintes:

- Assumir a existência de vários utilizadores que concorrem entre si para um mesmo leilão;
- Registar os vários leilões a decorrer;
- Realização de licitações (com a possibilidade de estabelecer um valor máximo de licitação);
- Alertar os licitadores quando o prazo do leilão estiver próximo do fim;
- Função que permite aumentar automaticamente o valor de licitação quando o prazo estiver a terminar;

O cumprimento dos presentes objetivos terá em vista a consulta e atualização de informação persistente, tendo em consideração a concorrência de acesso, a segurança e o desempenho do sistema. Para tal, criaram-se vários *scripts* que permitem a sua utilização num ambiente de sistema partilhado.

2. Introdução teórica

2.1. Linguagem de Programação C

2.1.1. Estruturas de dados

A estruturas de dados são utilizadas para permitir armazenar um conjunto de variáveis agrupadas num único identificador. Desse modo, é possível agrupar de forma organizada diferentes tipos de dados, acedendo às suas variáveis através de apontadores. De um modo genérico, as estruturas de dados, *structs*, são agrupadas da seguinte forma:

```
struct <identificador-da-estrutura>{
<tipo-de-dados> <identificador 1>;
<tipo-de-dados> <identificador 2>;
...
<tipo-de-dados> <identificador N>;
};
```

o `<identificador-da-estrutura>` representa o nome atribuído ao conjunto das variáveis; a declaração das mesmas é realizada em `<tipo-de-dados> <identificador N>`.

2.1.2. Ficheiros

Na linguagem C os ficheiros são sequências de *bytes* que podem ser acedidas de modo sequencial, aleatório ou direto.

Quando se acede a ficheiros em modo aleatório ou direto, acede-se por uma ordem aleatória a ficheiros binários com um número de *byte*s fixo. Uma vez que a dimensão destes ficheiros é fixa, eles não têm uma marca de "fim de elemento" porque o seu fim já é conhecido. Recorre-se a este tipo de ficheiros para grandes quantidades de dados, alterados frequentemente, que requeiram acesso direto e aleatório.

Por outro lado, o acesso a ficheiros de texto é feito de modo sequencial, os elementos são acedidos em posições consecutivas. A informação é guardada na forma de linhas de texto separadas pelo elemento "\n". Estes ficheiros apresentam fácil acesso e interpretação pelo utilizador, contudo a quantidade de dados armazenados não é muito elevada e estes não sofrem muitas alterações. Para alterar a k-ésima linha de um ficheiro de texto, primeiro tem de se criar uma cópia do ficheiro com as k-1 linhas que antecedem a alteração e com as restantes linhas.

2.2. Programação Concorrente

2.2.1. Semáforos

Os semáforos são utilizados para controlar o acesso de *threads* a um determinado código, sendo constituídos por um contador e por um valor inteiro que armazena o estado do mesmo.

Permitem controlar o número de *threads* que podem atuar nos dados partilhados. Antes de iniciar a operação, a *thread* deve requisitar o acesso, operação P(*proberen)*, decrementando um valor na contagem do semáforo. Se o valor do contador for maior ou igual a zero, a *thread* tem permissão de acesso, caso contrário, é bloqueada. O desbloqueio ocorre quando é realizada uma outra operação, operação V(*verhogen*), que incrementa o contador do semáforo, permitindo o acesso a uma das em espera, e imediatamente é decrementado novamente.

3. Projeto

3.1. Includes

Na linguagem C, todas as linhas que iniciam com # são diretrizes para o pré-processador, que significa que estas diretrizes serão processadas antes que o código seja realmente compilado. Neste trabalho recorreu-se à diretriz #include. Esta diretriz pode ter duas formas:

```
#include <header>
#include "file"
```

A primeira forma apenas é utilizada para incluir bibliotecas *standard*. Esta permite aceder a todas as funções que a biblioteca contem. A biblioteca *stdio.h* permite operações de *output/input*, como é o caso de operações de leitura de dados e exposição de informações no ecrã do programa. Relativamente à biblioteca *stdlib.h* contém funções relativas à alocação de memória, controlo de processos e conversões. Já a biblioteca *string.h* possui funções que permite a manipulação de cadeiras de caracteres e regiões de memória.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

A segunda forma da diretriz #include trata-se em substituir os cabeçalhos por ficheiros. Normalmente este tipo de forma é utilizado quando se pretende utilizar funções de um outro programa.

3.2. Structs

3.2.1. Struct Leilao

Neste trabalho desenvolveu-se a estrutura Leilao. A estrutura *Leilao* inclui a identificação do item (*idItem*), o preço mínimo do item (*price*), a data de inicio do leilão (*StartDateTime*), a data de fim do leilão (*EndDateTime*) e a identificação do utilizador (*idutilizador*), que neste caso se trata de quem irá comprar o item, e identificação do vendedor(*vendedor*). Esta estrutura inclui ainda o i*sOpen* que só pode conter dois valores zero ou um. Quando i*sOpen* é igual a 0, significa que o leilão ainda não está ativo, ou seja, a data atual ainda não ultrapassou a data de inicio do leilão, ou a data atual já ultrapassou a data de fim do leilão. Quando *isOpen* é igual a 1 significa que o leilão já está ativo e assim o utilizador já pode fazer licitações.

É de salientar que tanto o preço como o *idUtilizador* alteram-se a cada licitação efetuada pelo utilizador para um determinado *item.*

```
typedef struct Leilao{
int idItem;
double price;
int StartDateTime;
int EndDateTime;
int IdUtilizador;
int vendedor;
int isOpen;
}leilao;
```

3.3. *Scripts*

3.3.1. Leiloes

Este *script* pretende-se simular leilões, ou seja, criar aleatoriamente um determinado número de leilões e escrevê-los, através da função *fwrite*, num determinado ficheiro, neste caso, no ficheiro "fileleiloes.dat". A extensão do ficheiro usada foi a DAT, normalmente utilizada para armazenamento de dados. Esta é definida como sendo um

tipo de arquivo que é criado por uma aplicação de software e pode conter texto, imagens, vídeos entre outros tipos de dados. Tem ainda a caraterística de poder ser utilizado como um ficheiro *backup* para outros tipos de artigos.

Para efetuar esta simulação implementou-se um *fork*, ou seja, a simulação foi efetuada por um processo-filho. Gerou-se um único processo-filho que realiza o ciclo vinte vezes, gerando vinte leilões.

Também se garantiu que a identificação do item gerada é única e que o leilão inicialmente se encontra inativo, sendo necessário efetuar o script *geraleiloes* para o ativar (caso verifique as condições de ativação).

3.3.2. gera_leiloes

Com este *script* pretende-se que a partir do parâmetro existente*, isOpen* (0 ou 1) se consiga ativar ou não o leilão, através da comparação das datas de inicio e fim com a data atual. Neste trabalho consideraram-se as datas como valores inteiros, sendo a data atual o valor fixo 6.

Caso a data atual seja inferior (ou igual) à data final e superior (ou igual) à inicial, o leilão encontra-se ativo, ou seja, o *isOpen* toma o valor 1. Para alterar o valor do isOpen no ficheiro, recorreu-se às funções *fseek* (permite realizar procuras e acessos no ficheiro), *fread* (permite ler do ficheiro) e *fwrite* (permite escrever no ficheiro).

É ainda de referir que a atualização do ficheiro não é automática, ou seja, por cada atualização requerida tem de se efetuar este *script.*

3.3.3. Inserir

Este *script* permite que um vendedor adicione um leilão ao sistema. Nesta inserção a identificação do utilizador é 0, já que ainda não existem licitadores.

3.3.4. PrintLeiloes

Este *script* permite imprimir os leilões presentes no ficheiro (*fileleiloes.dat*).

3.3.5. Licitacao1

Este *script* foi introduzido para se realizarem licitações, introduzindo a identificação do item, o preço a licitar e ainda a identificação do licitador. É de salientar

que nas situações em que o preço da licitação for inferior ao preço atual do item, esta não será tida em conta.

3.4. Concorrência

Os processos de modificação do ficheiro implicam problemas de concorrência, sendo necessário sincronizar o acesso ao mesmo quando mais do que um utilizador tenta acedê-lo. Para tal recorreu-se à implementação de semáforos com valor inicial 1.

Desse modo, quando um utilizador X tenta efetuar alguma função controlada por semáforos, se já existir um utilizador Y a utilizar a mesma, o utilizador X ficará em espera até o utilizador Y terminar a sua tarefa.

No presente trabalho implementaram-se semáforos nos scripts: *inserir* (função *inserirleilao*), *licitacao1* (função *alterar*), *leiloes* (função *inserirleilao*) e no *gera_leiloes*.

4. Conclusão

A realização do presente trabalho permitiu aprofundar os conhecimentos relativos à linguagem de programação C. Ao longo da sua realização, surgiram problemas como por exemplo, a licitação remota através de *ssh* e a licitação automática.

Deste modo, os objetivos do trabalho foram parcialmente cumpridos. Assim sendo, desenvolveu-se um sistema de leilões em que é possível implementar um simulador de leilões, inserir leilões, torná-los ativos, licitar e imprimir os leilões.

5. Referências Bibliográficas

[1] *C development on Linux*: https://linuxconfig.org/c-development-on-linux-introduction-i (acedido pela última vez em 2017-12-19)

[2] *Synchronizing Threads with POSIX Semaphores*: http://www.csc.villanova.edu/~mdamian/threads/posixsem.html (acedido pela última vez em 2017-12-21)