Sistemas Operativos (2º ano de Mestrado Integrado em Engenharia Informática)

Trabalho Prático

Relatório

João Queirós (a82422)
José Santos (a84288)
Pedro Gomes (a84220)

23 de Julho de 2020

Conteúdo

Capítulo 1

Introdução

Este trabalho consiste no desenvolvimento de um sistema de gestão de vendas, constituído por várias entidades, de modo a simular a venda, aquisição e manutenção de diversos artigos.

O trabalho foi desenvolvido na linguagem C, tendo em mente a utilização das várias system calls presentes em ambiente Linux abordadas durante a duração da Unidade Curricular. O trabalho foca-se ainda numa gestão eficiente e escalável da memória principal, tirando partido da utilização de ficheiros, inclusive durante a sua execução.

1.1 Entidades do Sistema

1.1.1 Servidor de Vendas

O Servidor de Vendas(SV) é responsável pelo controlo do stock e pelo registo das vendas/compras efetuadas. Este recebe instruções dos vários Clientes assim como da Manutenção de Artigos, posteriormente este consulta e/ou altera os dados relativos aos produtos afetados pelas instruções recebidas. Durante a realização do trabalho, assumimos que apenas existe no máximo uma instância do servidor a ser executada, de modo que não realizamos testes referentes a potênciais problemas de concorrência derivados da existência de vários servidores em simultâneo.

O Servidor começa por criar o Pipe Principal, por onde vai receber as intruções provenientes dos clientes. Cada instrução é processada por um processo filho que executa a instrução, atualizando os ficheiros quando necessário, e envia a resposta ao cliente através do pipe apropriado.

1.1.2 Cliente de Vendas

O Cliente de Vendas(CV) interage com o SV, solicitanto a execução de instruções distintas entre si pelo número de parâmetros de cada instrução. As instruções existentes correspondem á consulta da informação pertinente a um artigo (o seu preço e o número de unidades em stock) ou á execução de uma entrada ou saída de stock relativa a um artigo (compra ou venda). O SV é capaz de suportar e satisfazer vários clientes ao mesmo tempo, pelo que as intruções requeridas por uma instância não afetam a realização das restantes.

Na nossa implementação do Cliente optamos por utilizar dois processos, um que lê as intruções e as encaminha para o Servidor, e um segundo que lê do Pipe para onde são enviadas as respostas obtidas pelo Servidor. Para tal, e como mencionado no ponto anterior, criamos um processo filho, sendo que o pai fica encarregue de receber as instruções do *stdin*, enquanto que o filho recebe as respostas e as envia para o *stdout*.

1.1.3 Manutenção de Artigos

A Manutenção de Artigos(MA) é responsável pela inserção de novos artigos no sistema, assim como pela potencial alteração dos nomes e preços de artigos existentes, alertando o SV para potenciais alterações de dados que possam existir em memória central. Tal como com o SV, assumimos que apenas haveria no máximo uma instância deste programa em execução, pelo que não realizamos testes com várias ocorrências de MA.

1.1.4 Agregador

O agregador(AG) é um programa que não interage diretamente com os outros, em vez disso, este é chamado pelo SV e pelo MA para permitir agregar o ficheiro de vendas. O AG lê a informação existente no ficheiro de vendas, e junta todas as entradas num só ficheiro com o mesmo formato, juntando no entanto todas as entradas correspondentes a cada artigo numa só.

Capítulo 2

Especificação e Implementação

Neste capítulo iremos especificar de uma forma mais aprofundada o funcionamento do sistema, assim como a sua implementação.

2.1 Especificação

2.1.1 Dados

Artigos

Ficheiro que contém o preço de cada artigo e um offset correspondente á posição do nome do artigo no ficheiro *Strings*. Cada artigo contém um código, que se encontra implícito no ficheiro, visto que cada entrada no mesmo possui tamanho fixo, sendo que o código do artigo corresponde ao seu índice no ficheiro.

Vendas

Ficheiro que contém os registos de cada operação de compra/venda efetuada pelo servidor. Cada entrada contém o código do artigo, o número de unindades referentes á compra, e o montante da mesma.

Stocks

Ficheiro que contém a quantidade em stock de cada artigo, também ordenados pelo código implícito.

Strings

Ficheiro que contém as strings correspondentes aos nomes dos vários artigos. Estes não se encontram ordenados, sendo a sua posição referida no ficheiro Artigos.

2.1.2 Comunicação entre as entidades do sistema

Uma das primeiras decisões com que nos deparamos foi a forma como os diversos programas deveriam comunicar durante a sua execução, começando pelas interações Cliente(s)/Servidor. Tendo em conta que o sistema deve suportar a existência de um servidor e vários clientes, decidimos que a forma mais simples e eficaz de permitir ao servidor receber as instruções dos múltiplicos clientes era através de um *Named Pipe* principal, de nome conhecido pelo servidor e clientes, criado aquando da execução do servidor.

Os clientes enviam instruções na forma de uma estrutura de tamanho fixo para o pipe, intruções estas que são lidas pelo servidor, que, ao ler uma instrução, executa um *fork*, criado assim um processo filho que por sua vez processa a instrução recebida. A estrutura utilizada possui o pid do processo Cliente que a enviou, sendo que após a execução da instrução, o servidor envia os resultados na forma de uma outra estrutura através de um outro Pipe criado pelo cliente com o nome igual ao seu pid, permitindo assim que todos os clientes enviem as instruções para o servidor, mas que as respostas ás mesmas sejam redirecionadas apenas para o Cliente as enviou. Criamos ainda um handler para *SIGINT*, de modo a que quando os programas Cliente e Servidor são terminados com *CTRL+C*, os respetivos pipes sejam eliminados.

2.2 Estruturas Utilizadas

Action

Estrutura utilizada para enviar instruções de clientes para o Servidor.

Contém o pid do cliente que enviou a instrução, o codigo do artigo e a quantidade a comprar/vender. Caso a quantidade seja 0, significa que a instrução é de consulta.

```
struct action{
    pid_t pid;
    int codigo;
    int quantidade;
};
```

Answer

Estrutura utilizada para permitir ao servidor enviar as respostas ás instruções dos clientes. Contém o número de artigos em stock e o seu preço.

Caso a instrução seja de consulta, o preço é devolvido como 0, e caso o artigo não exista retorna com valor igual -1, sendo que para isto, assumimos que não existem artigos com valores de preço nulo ou negativo.

```
struct action{
    int stock;
    int preco;
}
```

Artigo

Estrutura utilizada para tratar de entradas no formato do ficheiro Artigos. Contém o código do artigo, o offset do seu nome no ficheiro Strings, e o seu preço.

```
typedef struct _artigo{
    int ID;
    off_t stringRef;
    int price;
}Artigo;
```

Stocks

Estrutura utilizada para atualizar o ficheiro de stocks. Contém a código do artigo e a sua quantidade

```
typedef struct stocks{
 int numCod;
 int qnt;
}Stocks
```

Sale

Estrutura utilizada para tratar de entradas no formato do Ficheiro de Vendas. Contém o código do artigo, a quantidade referente á venda/compra, e o montante.

```
typedef struct sale{
    int ID;
    int qnt;
    int price
}Sale;
```

Cache

Estrutura que contém a informação guardada em cache relativa a um artigo. Contém o código do artigo, o preço, e o número de operações realizadas sobre o mesmo.

```
struct _cache{
    int ID;
    int price;
    int acessos;
}
```

File

Estrutura usada para guardar em ficheiro a informação relativas ao número de acessos de um artigo. Contém o código do artigo e o número de acessos ao mesmo.

```
struct file{
    int codigo;
    int acesso;
}File;
```

2.3 Funcionalidades implementadas

Para além do mencionado acima, o sistema contém ainda alguns aspetos mencionados pelos docentes, das quais:

2.3.1 Caching de Preços

O servidor contém um array de elementos Cache (contêm o código, preço, e números de acesso de um dado artigo). Quando é necessário executar uma instrução num artigo, o servidor começa por verificar se

este existe na cache. Caso não exista, procura no ficheiro artigos. Quando é executada uma instrução sobre um artigo, é incrementado o número de acessos ao mesmo. Caso este esteja na cache, é incrementado aí, caso contrário é incrementado no ficheiro fcache, que contém o número de acessos dos artigos ordenados pelo seu código implícito. Após incrementar no ficheiro, é verificado se o número de acessos é superior ao de algum dos artigos em cache, e caso seja, o novo artigo passa para a cache e o que tiver menos acessos é removido.

2.3.2 Agregação concorrente

O servidor divide o ficheiro de Vendas e cria processos filhos para cada divisão, sendo que cada processo filho processa apenas a parte do ficheiro que lhe é atribuída. No final a informação é agregada para um só ficheiro com timestamp da hora a que a agregação foi realizada.

2.3.3 Compactação do ficheiro Strings

O ma verifica se o tamanho do ficheiro de artigos é 20% maior que o de strings e se for realiza a compactação. A compactação passa por um varrimento do ficheiro de artigos , indo buscar a respetiva string de um artigo e pôr num ficheiro temporário, atualizando o offset na estrutura do artigo com o offset onde foi escrito no ficheiro temporário. No final o nome desse ficheiro temporario muda para strings(nome inicial).

Capítulo 3

Conclusões, Dificuldades e Alternativas

Ao longo do desenvolvimento do trabalho, o grupo deparou-se com algumas decisões em termos de implementação, sendo uma das primeiras a forma de comunicação entre as várias entidades do programa, nomeadamente Servidor/Cliente. Consideramos para isso a utilização de sinais para enviar as respostas do Servidor para o Cliente. No entanto, isto resultou em vários problemas, nomeadamente o sobreposicionamento de Sinais recebidos pelo cliente. Eventualmente decidimos utilizar somente Pipes para as interações, e tomando partido da utilização de estruturas de dados com tamanhos fixos, esta forma de implementação mostrou ser bastante prática e eficaz, pelo que decidimos que seria a melhor para o trabalho a ser desenvolvido.

Como os ficheiros já tinham sido delineados no enunciado pelos docentes da Unicade Curricular, a estruturação dos mesmos mostrou-se ser bastante útil na forma como abordamos os problemas que foram aparecendo, sendo que o índice implícito dos artigos, assim como o tamanho fixo de cada entrada provaram ser uma mais valia para a realização do trabalho e algo a ter em conta em trabalhos futuros.

No geral, apreciamos de forma positiva a forma como o grupo abordou o trabalho, estando satisfeitos com o resultado final, sendo que conseguimos implementar as funcionalidades descritas e aprendemos novas formas de abordar problemas e funcionalidades que antes da Unidade Curricular eram para nós desconhecidas.