Sistemas Operativos (2º ano de MIEI)

Trabalho Prático

Relatório de Desenvolvimento

Ricardo Pereira (a77045)

Rafaela Rodrigues (a80516)

Luis Duarte (a81931)

5 de Junho de 2019

Resumo

Este relatório inicia-se com uma breve contextualização, seguindo-se a declaração dos objetivos deste trabalho e em que é que este consiste.

De seguida é apresentada a análise feita ao enunciado, que será a base a partir do qual é desenhada a resolução final, que é enunciada seguidamente. Na conceção da resolução, são apresentadas as estruturas de dados utilizadas, bem como a implementação da resolução.

Findo esse capítulo, são apresentados alguns problemas de implementação e decisões tomadas para os resolver. Depois surgem os testes e resultados produzidos.

Por fim, são apresentadas conclusões sobre o trabalho desenvolvido.

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

O presente relatório foi elaborado no âmbito do Trabalho Prático da Unidade Curricular de Sistemas Operativos, que se insere no 2º semestre do 2º ano do primeiro ciclo de estudos do Mestrado Integrado em Engenharia Informática.

1.2 Objetivos e Trabalho Proposto

Para este trabalho foi proposta a elaboração de um protótipo de um sistema de gestão de inventário e vendas, consituido por quatro programas diferentes, sendo eles o manutenção de artigos, agregrador, servidor e cliente vendas. estes programas serão capazes de representar um sistema real que cumpre os pré-requisitos e funcionalidades propostas, e ainda demontram as principais vantagens de um sistema operativo. Estas vantagens passam por exemplo pela abstração de hardware feita pelo sistema operativo, permitindo o funcionamento do software em qualquer computador, ou ainda a gestão das tarefas e trabalho executado pelo computador permitindo que vàrios programas estejam a ser executados em simultâneo.

De seguida é apresentado o raciocinio feito por este grupo de trabalho para a concepção do sistema proposto.

Capítulo 2

Conceção/desenho da Resolução

Para a conceção da solução para o problema proposto, começamos por identificar quais os dados e estruturas necessários para o funcionamento do sistema e a estratégia usada para a implementação de cada funcionalidade.

2.1 Ficheiros de Dados

De acordo com o enunciado do projeto, foram criados e utilizados os ficheiros artigos, strings, stocks e vendas.

Dado que não foi específicado o tipo de ficheiro (binário,texto), os ficheiros artigo e stocks são binários, para facilitar a sua consulta.

No entanto, existe um programa denominado imprimir que apresenta todo o conteúdo dos ficheiros referidos, em forma de texto. Nas secções seguinte será explicitado o conteúdo guardado em cada ficheiro.

2.2 Estruturas de Dados

Com o objetivo de simplificar a recolha de dados e o seu armazenamento, foram usados 2 tipos de estruturas de dados auxiliares:

- Artigo;
- Stock;

A struct Artigo permite-nos guardar toda a informação referente a um artigo, ou seja, o seu código (identificador), posição do nome do artigo no ficheiro **strings** e o seu preço.

Todas as estruturas referentes aos artigos adicionados serão guardadas no ficheiro **artigos**.

A estrutura stock permite guardar o código de um artigo e o seu stock. Estas estruturas serão armazenadas no ficheiro stock.

2.3 Manutenção de artigos

O programa manutenção de artigos tem como principal função a inserção de novos artigos no sistema desenvolvido, a alteração de preço ou nome de artigos pré-existentes e ainda a chamada do programa

agregador. Desta forma, este programa manipula os ficheiros mediante instruções que recebe, nomeadamente, o ficheiro artigos(ficheiro com a informação relativa aos artigos do sistema) e o ficheiro strings(ficheiro que contém os nomes dos artigos existentes no sistema).

Este programa recebe todas as instruções a executar pelo seu input e depois do seu tratamento, isto é, ver que tipo de instrução se trata(por exemplo se é uma inserção de um artigo), executa-a. Nas subseções seguintes é explicado o algoritmo usado para o tratamento das diferentes intruções que este programa recebe de modo a cumprir com os objetivos acima referidos.

2.3.1 Insersão de artigos

Caso a instrução passada através do input ao programa seja do tipo "i nome preço", o programa terá que inserir no ficheiro artigos uma nova entrada.

A estratégia usada para responder a esta necessidade foi a seguinte: Primeiramente é atríbuido o código que corresponderá ao código do novo artigo a ser inserido (é sequencial). Para isto , é calculado o tamanho total do ficheiro artigos, e como cada artigo tem um tamanho fixo, o calculo do código é direto.

De seguida após abertura dos ficheiros artigos e strings, é feito o posicionamento dos respetivos descritores no final de cada ficheiro de modo a que a inserção da nova informção seja feita nos sitios corretos. Graças ao posicionamento do descritor do ficheiro strings é ainda possível determinar a posição do nome do novo artigo.

Após este conjunto de operações é criado um então novo artigo com o código e posição calculados anteriormente, e é feita a escrita deste no ficheiro artigos, sendo o seu nome escrito no ficheiro strings.

Denotar que cada vez que é feita uma abertura de um ficheiro é feito também o seu fechamento de modo a preservar um correto funcionamento do programa.

2.3.2 Alteração de informação

A alteração de informação pode ser de dois tipos distintos. Caso a instrução recebida pelo programa seja do tipo "n código nome"o programa deverá alterar o nome do artigo. O outro tipo de instrução é "p código preço", e neste caso o programa deve alterar o preço do artigo.

O algoritmo usado para a alteração de um nome consiste em: Novamente começa-se por abrir os ficheiros que serão manipulados, sendo neste caso o ficheiro artigos, onde a posição do artigo cujo nome está a ser reposto será atualizado, e o ficheiro strings, onde se feita a inserção do novo nome.

Para isto, graças ao código do artigo é feito um posicioamento do descritor do ficheiro artigos na posição onde está o artigo que terá a sua posicção alterada, e feito um posicionamento do descritor do ficheiro strings no fim do ficheiro(Este ultimo posicionamento retorna ainda a nova posição do nome do artigo no ficheiro strings). Após esta operação é lido então o artigo em causa do ficheiro artigos, altera-se a sua posição pela posição calculada previamente e este é escrito na posição de onde foi lido. Por fim é escrito o novo nome no final do ficheiro strings.

A fórmula implementada para o tratamento do outro tipo de instrução, nomeadamente para a alteração de um preço foi a apresentada de seguida: Começa-se por abrir o ficheiro artigos(único manipulado neste tipo de operação) e através do código é feito o posicionamento do seu descritor. Apoś isto, é lido o artigo em causa, é alterado o seu preço e inserido na mesma posição.

De realçar novamnete que cada vez que é feita uma abertura de um ficheiro esta é acompanhada pelo respetivo fechamento.

2.3.3 Chamada do agregador

O ultimo tipo de operação associado ao programa manutenção de artigos é a invocação do agreagador.

Desta forma a solução implementada passa por cada vez que o programa recebe uma instrução do tipo "a"despoletar um sinal personalizado para o pid do processo do programa servidor de vendas (que se encontra a correr). Este pid é escrito previamente num ficheiro auxiliar pelo servidor de vendas e posteriormente lido pelo manutenção de artigos. O tratamento do sinal é feito no servidor de vendas.

2.3.4 Compactação do ficheiro STRINGS

De forma a reduzir o desperdício de espaço, sempre que o ficheiro strings superar os 20% de espaço ocupado por nomes obsoletos é feita uma compactação para um novo ficheiro e são ajustadas as posições no ficheiro de artigos.

A estratégia utilizada começa por definir como e quando é efetuada esta compactação. Guardamos duas variáveis globais, "tamanhoDesperdicado"e "tamanhoTotal", que corresponem respetivamente ao tamanho desperdicado no ficheiro e ao tamanho total do mesmo. Sempre que é efetuada uma alteração de nome, é feito um calculo (tamanhoTotal/tamanhoDesperdicado) e se este for maior que 0,2 então é necessário executar uma compactação do ficheiro.

O algoritmo é bastante simples, é criado inicialmente um ficheiro novo, "stringsv2", onde serão escritos os nomes atuais dos artigos. É percorrido o ficheiro artigos, e a cada artigo vamos buscar a posição onde está escrito o seu nome no ficheiro strings, esse nome é lido através de uma função auxiliar que recebe essa mesma posição e coloco num apontador passado como argumento pelo compactador o nome do artigo em questão. De seguida, é escrito o nome no novo ficheiro strings, e calculada a posição onde foi escrito, assim inserimos novamente o artigo no ficheiro artigos, com a informação atualizada sobre a posição do seu nome.

No final é eliminado o ficheiro antigo e renomeado o novo ficheiro para "strings". De forma a que quando o ma for executado novamente tenha informação do estado atual do ficheiro strings quanto ao espaço desperdiçado, as duas variáveis globais, "tamanhoDesperdicado"e "tamanhoTotal"são escritas num ficheiro "tamanho"antes do fecho da execução do programa e carregadas sempre do ficheiro para o programa no seu início.

2.4 Cliente de vendas

O cliente de vendas é um programa que tem o objetivo de enviar pedidos ao servidor e receber as respostas do mesmo.

Com o intuito de identificar de qual cliente de venda que enviou o pedido, dado que todos são enviados ao servidor pelo mesmo pipe, foi decidido juntar ao fim da instrução o pid do processo (cliente), na forma "instrução:pid". Este pid também irá fornecer o nome ao pipe que permitirá receber as respostas do servidor.

Então, o cliente recebe pelo input um conjunto de instruções. Após ser adicionado o pid, o cliente envia pelo pipe do servidor a instrução e fica à espera da resposta, escrevendo o conteúdo recebido para o seu output.

Por fim, é apagado o pipe criado pelo cliente e a sua execução termina.

2.5 Servidor de vendas

O servidor de vendas é responsável por controlar os pedidos provenientes do cliente de vendas (inserção de vendas/stock e consulta de preços) e correr o agregador quando pedido.

2.5.1 Pedidos cliente de vendas

Como dito anteriormente, o pedido de vendas é responsável por executar os pedidos proveniente dos clientes. Dado que é necessário que o servidor esteje sempre pronto a receber pedidos dos demais clientes, foi desenhado o seguinte algoritmo:

No momento em que se executa o servidor, e sabendo que este será o primeiro a ser executado, é criado um pipe com nome (com nome pré-definido) que ficará à espera para ler pedidos. Por sua vez, os clientes escrevem para esse mesmo pipe todos os pedidos, como dito anteriormente.

De forma a permitir a execução concorrente de clientes de vendas, é necessário impedir que o mesmo artigo seja acedido por diferentes processos (para evitar inconsistências e erros). Para tal, foram criados 10 processos (filhos) e 10 pipes anónimos. Cada filho fica responsável por ler a informação proveniente de um pipe e executar os pedidos recebidos.

Como queremos evitar que os filhos acedam á mesma informação, seja relativa aos stocks ou aos artigos, a estratégia utilizada foi utilizar o código de cada pedido e calcular o módulo para 10 (código%10). Este cálculo fornece-nos a parte decimal, que irá de 0 a 9. Então, esse pedido é encaminhado para o pipe identificado por esse resultado. Assim todas as instruções de uma determinada gama vão para o mesmo processo, garantido exclusão mutúa.

Após a instrução ter sido encaminhada para o processo correto, este processa o pedido e escreve o resultado para um pipe anónimo, onde do outro lado estará um filho que irá escrever o resultado para o pipe que liga o servidor ao cliente que enviou o pedido.

2.5.2 Venda de stock

O pedido para efetuar uma venda é encaminhado para uma função que irá processar a mesma, quando a quantidade da instrução é negativa. Esta função recebe o código e quantidade de venda.

Primeiramente é aberto o ficheiro stocks e posicionado o descritor do ficheiro para a posição do stock respetivo ao código recebido.

De seguida, é efetuada uma leitura do stock pelo que, se nada for lido significa que ainda não havia nenhuma entrada de stock para o artigo em questão, nesse caso a função retorna -1 e o é enviada uma resposta ao cliente informando que não há stock disponivel.

No caso de haver stock, é verificado se o valor de stock disponivel é maior que o valor requerido para venda, se sim a venda é efetuada normalmente e é retirado ao stock o valor da venda, e por sua vez é inserida a venda no ficheiro vendas. Se o stock disponivel for menor que a quantidade requerida, é efetuada a venda de todo o stock disponivel e inserida a venda no ficheiro vendas.

No final, é sempre retornado ao cliente o valor do novo stock.

2.5.3 Adição de stock

Tal como na situação anterior, o pedido para adicionar stock é reencaminhado para uma função designada para a adição do mesmo. Esta função recebe o código do artigo correspondente e a quantidade a adicionar.

Primeiramente, é lido do ficheiro stocks o stock correspondente ao artigo. No caso de não ser lido nenhum stock, é a primeira inserção de stock relativo ao artigo em questão. É criado um stock com a quantidade recebida como argumento e inserido esse stock no ficheiro stocks. No caso de haver stock, é adicionada a quantidade recebida ao mesmo e inserido novamente o stock no ficheiro.

No final é retornado o stock atual para a função que reencaminhou a instrução para a adição de stocks, que por sua vez envia uma resposta ao cliente informando da nova quantidade de stock.

2.5.4 Consulta de preço e stock

Quando o cliente envia uma instrução apenas com o código, é requerida uma consulta de preço e stock atual de um artigo.

Para isso, utilizamos duas funções, a primeira faz uma consulta ao ficheiro artigos para saber o preço e a segunda uma consulta ao ficheiro stocks para saber o stock.

No final é retornada ao cliente uma resposta com o valor atual do preço e do stock.

2.5.5 Caching de Preços

De forma a minimizar os acessos ao ficheiro artigos para a consulta de preços, foi implementada uma cache de artigos.

Esta consiste num array "Artigo cache[10]"onde serão guardados os últimos 10 artigos consultados para saber o seu preço. Para isso guardamos mais duas variáveis, o número de artigos atual guardados na cache e a posição atual de inserção na cache.

O número atual de artigos em cache, serve para evitar que enquanto a cache não está cheia não é procurado um artigo numa posição não preenchida no array. A posição atual da cache é utilizada sempre que se adicionar um artigo á cache, o artigo é adicionado nessa posição e a posição atual é incrementada 1 valor caso não esteja já na última posição, se estiver na última posição a valor atribuido á variável é 0, de forma a tornar o array de cache, num array FIFO.

A cache é utilizada sempre que é preciso saber o preço de um artigo, para isso criamos uma função que recebe o código de um artigo e retorna, -1 caso o artigo não esteja na cache, o valor do preço do artigo caso esteja na cache.

No caso de não estar na cache, é feita a consulta preço recorrendo ao ficheiro artigos, e por sua vez, a inserção desse artigo na cache, em deterimento do primeiro artigo colocado em cache atualmente.

2.5.6 Pedido de agregação

O pedido de agregação é enviado ao servidor pela manutenção de artigos, através de um sinal.

Quando o servidor recebe esse sinal, é executada a função agrega.

Para a agregação ser mais eficiente, decidimos criar concorrência. Dado o intervalo para a agregação, esse intervalo é dividido no máximo por 4 filhos (caso hajam menos de 4 linhas para agregar, são criados tantos filhos quantas linhas). Cada filho executa uma instância do programa ag (agregador) e recebe pelo pai todas as vendas a agregar (este lê do ficheiro vendas e reencaminha aos filhos por pipes anónimos).

O resultado dessas agregações são enviadas a outra instãncia do ag (executado por outro filho) através de um pipe e este escreve o resultado da agregaçãi das agregações anteriores para um ficheiro cujo nome é a data e hora em que a agregação foi pedida.

Após a agregação estar terminada, o servidor volta para o estado onde parou.

2.6 Agregador

O agregador tem como objetivo agregar as informações das vendas de acordo com o artigos, isto é, dado um intervalo (recebido pelo input), para cada artigo são somados todos as quantidades compradas e preço total, devolvendo pelo output o resultado da agregação.

Ao íncio, o plano para guardar as agregações era escrever o resultado das sucessivas agregaçõe num ficheiro, e depois percorrer o resultado e escrevê-lo para o output do programa.

No entanto, devido a um erro que não foi possível corrigir e o curto tempo disponível, optamos por guardar num array os resultados.

Sabemos que esta solução não é a melhor, devido ao limite que a memória possui, mas foi a solução alternativa encontrada.

Em cada valor da agregação é guardado o resultado do respetivo index. Se já existir o resultado da agregação, este é adicionado à nova agregação e guardado novamente. Caso não exista, é adicionado no array.

De ressalvar que os dados de input e output respeitam o formato do ficheiro de vendas.

Capítulo 3

Análise de Resultados

3.1 Testes efetuados

De forma a confirmar a implementação e a verficar o correto funcionamento do programa, efetuamos alguns testes no que consiste a cada um dos elementos fundamentais do programa.

No que diz respeito á manutenção de artigos, e por sua vez, ao capítulo da compactação do ficheiro strings, construímos exemplos de scripts. Verificamos o correto funcionamento, tanto na manutenção de artigos como na compactação do ficheiro strings. Corremos um script com diversas instruções, com 12Kb de dados diversas vezes, e verificou-se tanto a integridade dos dados como a eficiência do programa.

3.2 Problemas de implementação

Infelizmente, não conseguimos implementar todas as funcionalidades desejadas.

No momento em que passamos o pedido do cliente de vendas para o servidor através do pipe, nomeadamente os pedidos de consulta de preço e stock, a informação recebida do pipe não é o esperada e não conseguimos resolver.

Para além disso, o agregador deixou de funcionar no momento em que realizamos o último teste e devido ao facto que o trabalho era para ser entregue em pouco tempo, não conseguimos resolver.

Capítulo 4

Conclusão

Após a realização deste trabalho prático, sentimos que o grupo evoluiu no que concerne aos Sistemas Operativos e que de certa forma, conseguiu desenvolver um trabalho que responde a quase todos os requisitos.

Foram encontradas várias dificuldades, que foram sendo ultrapassadas por decisões de implementação discutidas e previamente analisadas para que fossem as melhores.

No final, tivemos alguns contratempos explicados no capítulo de testes, mas que esperamos resolver mesmo após a entrega do trabalho.