PROJETO - SISTEMAS OPERATIVOS

2019

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Grupo 16 – A83631, A85006, A84775

Filipa Santos, Hugo Cardoso, João Cunha



ÍNDICE

1. INTRODUÇÃO

O projeto de Sistemas Operativos consiste no desenvolvimento de um sistema gestor de vendas, que permite a inserção de novos artigos no ambiente, a mudança a qualquer momento dos seus atributos, nomeadamente o nome e o preço, e entradas e saídas de stock (vendas). Foi designado com um modelo de grande escala em mente, tendo por objetivo simular um tal sistema muito comum na sociedade moderna. Neste relatório, é explanada em detalhe a abordagem tomada por este grupo ao projeto, os problemas que surgiram pelo caminho e as soluções encontradas.

2. Ficheiros de texto

Toda a informação do gestor de venda é guardada em ficheiros de texto, nomeadamente em 4 principais – artigos, stocks, strings e vendas - e estão todos guardados numa pasta à parte denominada “Ficheiros”.

Todos, à exceção do ficheiro de vendas, têm linhas de comprimento fixo de maneira a facilitar a leitura e escrita nos mesmos. Como tal, definiu-se um certo número de bytes por elemento (código, preço, etc), tendo em conta que o objetivo do gestor é ser usado em escala industrial, podendo assim ser sujeito a milhares, senão milhões, de operações.

Foram portanto escolhidos valores apropriados, todos eles definidos no ficheiro header “cod_partilhado.h”. Esta definição de variáveis num ficheiro à parte, ao revés do uso dos valores “soltos” pelo trabalho, permite também a fácil alteração destes valores, caso surja a necessidade.

Artigos – cada linha tem o código, o preço e a posição do nome correspondente no ficheiro de strings.

Stocks – em cada linha está presente o código e o stock do artigo.

Strings – este é o único ficheiro que não tem um número fixo de bytes por linha. Está presente na primeira linha do ficheiro um contador do número total de bytes ocupados pelos nomes dos artigos no ficheiro, que como tal é atualizado a cada inserção nova. De resto, cada linha tem um nome escrito, podendo ser obsoleto ou não (caso tenha sido mudado o nome desse artigo).

Vendas – este ficheiro também possui um número no início que corresponde à posição da venda pela qual o gestor deve iniciar a próxima agregação, e todas as outras linhas têm comprimento fixo, possuindo o código do artigo vendido, o número de unidades vendidas e a receita gerada na venda.

3. Manutenção de artigos

A manutenção de artigos permite inserir novos artigos (especificando o nome e preço de venda) e alterar atributos de um dado artigos (nome ou preço).

Como o programa tem de receber todo o seu input pelo stdin, decidiu-se recorrer a um ficheiro de texto auxiliar chamado “comandos_ma”. Na compilação do programa através do respetivo comando da Makefile (“make run_ma”), o conteúdo deste ficheiro de texto é lido e redirecionado para o stdin (“./ma < comandos_ma”). Portanto, a fim de usufuir da manutenção de artigos é necessário escrever as instruções pretendidas nesse ficheiro.

3.1 Inserção de artigos

A inserção de um artigo é bastante simples. Caso seja a primeira inserção do programa, são criados os ficheiros de texto dos artigos, das strings e dos stocks.

Em seguida, o comando do utilizador é sujeito a uma validação, a fim de verificar que está bem escrito e evitar comportamento não definido da parte do programa. Esta validação passa por verificar se o código dado está dentro dos limites definidos, se a posição do novo nome não excede o número de bytes reservados para as mesmas e se o preço é, de facto, um número inteiro/decimal sem caracteres estranhos. Caso o comando seja inválido, é imprimida no stderr uma mensagem de erro e a operação de inserção não é realizada.

Após certificar que o comando está correto, escreve-se o nome do artigo novo no fim do ficheiro de strings e soma-se ao contador no início do ficheiro o número de bytes ocupados por esta string. Toma-se também nota da posição em que foi escrita o nome.

De seguida, adiciona-se a informação do artigo novo ao ficheiro de artigos, colocando numa linha nova e por esta ordem, o código, o preço e a posição do nome. Por fim, escreve-se também o código novo no ficheiro de stocks e inicializa-se por assim dizer o stock do novo artigo, que é nulo.

3.2 Mudança de nome

Ao mudar o nome de um artigo, o nome antigo não é apagado, sendo que simplesmente se acrescenta a nova string ao fim do ficheiro correspondente, atualizando depois o contador de bytes. A posição do nome no ficheiro dos artigos é também atualizada. Tudo isto, claro, se a instrução passar na validação.

3.3 Mudança de preço

Mudar o preço é a operação mais simples das três, sendo que só implica a edição do ficheiro de artigos. O programa valida o comando e navega até ao sítio do preço do artigo respetivo ao código e escreve por cima o novo preço.

4. Servidor de vendas

Este programa é responsável por controlar stocks, receber pedidos do cliente de vendas e registar as vendas efetuadas, e ainda correr o agregador a pedido. O utlizador não interage diretamente com o servidor, sendo as suas instruções encaminhadas para este pelo cliente de vendas ou pela manutenção de artigos. Possui ainda um sistema de caching de preços que será aprofundado mais adiante no relatório.

Ao ser inicializado, o servidor cria uma pipe pela qual recebe todas e quaisquer solicitações por parte dos outros programas e fica em ciclo infinito à espera de instruções.

4.1 Consulta de stock

Esta é uma das operações requisitadas pelo cliente de vendas e consiste em receber um código e devolver o stock atual e o preço do artigo correspondente. Quanto ao stock, o programa consulta o ficheiro de stocks e navega até ao código em questão para o ler. Já a estratégia para obter o preço é mais elaborada. Existe um sistema de caching que mantém os preços dos produtos mais populares em memória. Se o artigo já tiver sido consultado antes, existe a possibilidade de o seu preço estar guardado na cache, portanto o programa verifica se o código em questão existe lá e, em caso afirmativo, lê-o logo daí, evitando um acesso ao ficheiro dos artigos. Caso contrário seria isso mesmo que o programa faria, análogo à obtenção do stock.

É ainda pertinente apontar que, semelhante à manutenção de artigos, o servidor também valida os comandos do utilizador antes de os executar. Numa fase menos adiantada do projeto, verificava-se mesmo se o código dado correspondia ao código de algum artigo. Contudo, desistiu-se desta abordagem por ser muito custosa, dado que obrigava a acessos constantes ao ficheiro dos artigos, o que contraria o propósito do caching de preços. É portanto assumido que o utilizador introduz códigos válidos.

4.2 Alteração de stock

Esta já é uma operação mais complexa, que consiste em atualizar o stock segundo entradas do mesmo e vendas. Também ao stock é reservado um número fixo de bytes, sendo por isso recusadas quaisquer instruções com quantidades que excedam o limite estabelecido. No caso de uma entrada de stock, verifica-se se a soma do stock atual com a nova quantidade ainda está dentro dos limites e atualiza-se o stock com a mesma, caso esteja.

Já numa venda, um raciocínio análogo é usado, dado que não se pode vender mais unidades do que se tem em stock. Caso a quantidade demandada seja superior à quantidade em stock, não é efetuada venda nenhuma. Caso contrário, atualiza-se o stock no ficheiro, subtraindo-lhe o número de unidades vendidas, e regista-se no ficheiro de vendas a nova venda, com o código do artigo, a quantidade vendida e a receita gerada.

O servidor devolve então uma string com o resultado das operações ao cliente de vendas, que a mostra ao utilizador.

5. Cliente de vendas

O cliente de vendas interage com o servidor de vendas, solicitando-lhe a execução de operações que se distinguem facilmente pelo número de parâmetros introduzidos pelo utilizador. O cliente de vendas realiza esta comunicação com o servidor através de *named pipes*, dado que consistem em dois programas diferentes. Ao ser inicializado, o servidor cria uma pipe principal pela qual recebe todas as instruções do cliente. Este limita-se a reencaminhar a instrução introduzida pelo utilizador para o servidor, não fazendo a habitual verificação dado que isso é feito do outro lado.

O sistema permite também a execução concorrente de vários clientes de vendas, podendo o servidor estar em contacto com múltiplos ao mesmo tempo. De maneira a saber distinguir entre todos eles e não enviar as respostas para o cliente errado, cada cliente cria também uma pipe distinta. Para assegurar que dois clientes não tentam abrir uma pipe com o mesmo nome, cada um gera um processo filho ao executar um comando e anexa o pid desse processo ao nome da pipe, garantindo assim que todas as pipes são únicas, pois é impossível mais do que um filho ter o mesmo pid. Ao receber a resposta do servidor, o cliente simplesmente exibe-a ao utilizador e termina o processo filho gerado, apagando também a pipe criada pelo mesmo, de maneira a evitar que se acumulem.

Semelhante à manutenção de artigos, existe também um ficheiro de texto auxiliar chamado “comandos_cv” onde o utilizador deve escrever as instruções que pretende executar, sendo o conteúdo desse ficheiro depois redirecionado para o stdin do cliente de vendas aquando da despoletação do executável deste programa.

6. Agregador

O programa agregador funciona como filtro. Recebe pelo stdin entradas no formato do ficheiros de vendas, até end-of-file. Produz então para o stdout os dados agregados de cada artigos com vendas efetuadas. Este programa só conhece stdin e stdout, sendo portanto necessário recorrer a dups para redirecionar os mesmos de maneira a ser capaz de comunicar com os outros programas.

O agregador é convocado na manutenção de artigos, que envia o pedido ao servidor de vendas, dado que é este o responsável por correr o agregador. A manutenção comunica com o servidor pela pipe principal criada pelo mesmo para receber pedidos de outros programas. O servidor, por sua vez, invoca a execução do agregador através de um exec, criando para tal efeito um processo filho, de maneira a que a sua própria execução não seja interrompida. Como os programas iniciados a partir de execs herdam os descritores dos programas invocadores, o servidor cria uma pipe nova para comunicar com o agregador, abre-a e redireciona o stdin para a pipe. Tudo isto ocorre dentro do processo filho, por isso o stdin do servidor em si não é alterado.

O servidor abre então o ficheiro de vendas e navega até à venda mais antiga não agregada, através da sua posição que é guardada no início do ficheiro. Escreve todas as vendas daí até end-of-file na pipe e atualiza a posição inicial para end-of-file, onde irá ser inserida a próxima venda que se efetuar.

Devido ao redirecionamento anterior, o programa agregador recebe estas vendas pelo stdin e começa a processá-las. Verifica o código de cada venda e cria um ficheiro com o mesmo no nome, onde guarda todas as vendas desse artigo. Depois de fazer isto a todas, percorre todos os ficheiros um a um, somando as quantidades e as receitas individuais, e produz para o stdout os valores totais agregados de cada artigo. Porém, antes disso, redireciona-se o stdout para uma pipe, de maneira a o agregador conseguir comunicar com os outros programas.

Agora aqui há uma complicação. O objetivo era abrir outra pipe entre o servidor e o agregador e devolver pela mesma as vendas agregadas. Contudo, isto provou ser mais difícil do que inicialmente se pensou e não foi possível realizar esta arquitetura. A alternativa que se encontrou foi abrir uma pipe entre a manutenção de artigos e o agregador e enviar por aí os resultados, que a manutenção depois reencaminha para o servidor. Esta

solução, embora redundante e pouco eficiente, permite conservar a comunicação com outros programas solamente pelo stdin e stdout.

Por fim, o servidor recebe os resultados pela pipe principal e cria um ficheiro cujo nome reflete a data e a hora em que a agregação foi solicitada, escrevendo aí os mesmos.

7. Aspetos valorizados

7.1 Caching de preços

Esta é uma estratégia dedicada a evitar o acesso constante do servidor ao ficheiro de artigos para consultar os preços, mantendo em memória essa informação. Para este efeito, foi criada uma estrutura constituída por um array de códigos, um de quantidades e outro de preços. Ao arrancar, o servidor inicializa esta estrutura. A partir daí, à medida que o utilizador for requisitando serviços do servidor, este vai optando entre o ficheiro e a cache.

Considerou-se apropriado alocar 250 posições para cada array, o que continua a ser mínimo comparado à escala a que o gestor de vendas é usado. Enquanto os arrays não estiverem cheios, o servidor guarda os preços usados em qualquer operação, mesmo sendo uma consulta ou uma entrada de stock.

Contudo, o objetivo desta estratégia é manter em memória os preços dos artigos mais populares, visto que esses são requisitados com mais frequência. Daí o array de quantidades. Ao executar uma venda o servidor compara a quantidade vendida com as quantidades guardadas em cache (agora já com os arrays cheios) e, caso esta seja superior, substitui a informação do artigo menos vendido em cache pelo novo. À medida que o cliente for fazendo vendas do mesmo produto, vai-se adicionando à sua quantidade vendida, de maneira a ter uma noção dos artigos mais populares.

Assim, a cada operação que realiza, o servidor verifica se o preço que pretende já se encontra na cache, poupando potencialmente o tempo de

acesso e leitura do ficheiro, que repetido ao longo de milhares de operações acaba por se tornar muito custoso.

7.2 Agregação concorrente

Neste trabalho prático não foi implementada esta estratégia. A agregação ocorre de forma sequencial, sendo porém a solução implementada a mais eficaz em que se pensou. Houve tentativas de tornar a agregação concorrente, contudo falharam e decidiu-se por esta parte do trabalho de parte para se focar nas outras.

7.3 Compactação do ficheiro de strings

Como foi referido anteriormente, ao mudar o nome de um artigo a string antiga não é apagada, permanece simplesmente no ficheiro e é substituída a posição do artigo para a da nova string. Como é óbvio, isto pode levar a grandes desperdícios de espaço, o que propulsionou a ideia da compactação do ficheiro.

Sempre que os espaço ocupada por dados obsoletos chegar aos 20% do ficheiro total, é feita a compactação para um novo ficheiro e ajustadas as posições no ficheiro de artigos.

A implementação desta estratégia é bastante complicada e começa logo na mudança de nomes. É usado um ficheiro de texto auxiliar (“info”) onde se guarda informação acerca das strings inválidas.

A primeira linha possui um contador do número de bytes obsoletos no ficheiro de strings (que é mais tarde comparado com o contador do número total de bytes presente nesse ficheiro), a segunda linha possui um contador do número de strings inválidas e daí em diante tem-se as posições das strings inválidas. Ao mudar um nome, iteram-se os contadores deste ficheiro e anexa-se a nova posição invalidada ao fim do ficheiro.

Após a inserção do novo nome no ficheiro de strings, verifica-se se 20% do ficheiro já é ocupado por lixo ou não. Em caso afirmativo, inicia-se a limpeza do mesmo. Cria-se um ficheiro novo e copia-se para lá as strings válidas. Para esta cópia, aloca-se memória do para suportar toda a informação entre duas strings inválidas no ficheiro e copia-se todo esse conjunto de strings entre elas de uma só vez. Isto, embora muito eficaz, pois evita que se esteja a percorrer o ficheiro linha a linha e a verificar o que é válido ou não, tem as suas limitações, pois no caso de um número extremamente elevado de linhas entre duas strings inválidas, há o risco de a memória rebentar ao ser alocada tanta de uma vez. Vai-se então copiando pedaços do ficheiro para o novo até que no fim se apaga o ficheiro antigo e renomeia o novo de volta para strings.

Depois disso, percorre-se o ficheiro de artigos e atualiza-se as posições das strings. Tendo as posições onde se encontravam as strings inválidas e lendo o número de bytes de cada uma delas, vai-se comparando às posições dos artigos e, conforme a posição destas em relação às inválidas no ficheiro anterior, vai-se-lhes subtraindo o número de bytes que ocupavam os nomes removidos.

8. Conclusão

Em suma, o grupo está satisfeito com o resultado adquirido. Foi um projeto desafiante, embora relativamente curto, e ajudou a compreender mais a fundo a matéria de Sistemas Operativos e como implementá-la. Há algumas falhas no trabalho que com mais algum tempo seriam, à partida, resolvidas, mas apesar de tudo considera-se um bom esforço coletivo e uma boa prestação.