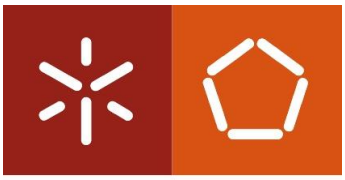

Universidade do Minho

Departamento de Informática

Gestão de Stocks

Grupo Nº 2

José Ferreira (A83683)

Emanuel Rodrigues (A84776)

João Teixeira (A85504)

12 de Maio de 2019

Conteúdo

Capítulo 1

Introdução

O objetivo deste projeto é construir um sistema de gestão de stocks, de forma a ser capaz de armazenar informação de artigos, stocks e vendas, aplicando uma arquitetura *Cliente, Servidor* com recurso a *System Calls* e vários processos, aplicando assim os conhecimentos sobre os Sistema Operativos baseados em Unix adquiridos nas aulas desta UC. Como objetivo, é necessário implementar um sistema que permita a utilização concorrente de um servidor por vários clientes, assegurando a conservação de toda a informação introduzida no sistema, bem como a consistência desta entre os vários ficheiros e cache.
Ao longo deste relatório vamos descrever as nossas abordagens a estes problemas.

Capítulo 2

Módulos

2.1 Manutenção de Artigos

O modulo da Manutenção de artigos tem como função ser um 'preçário' tendo informação dos diferentes produtos existentes, bem como o seu preço. Aqui é possível a criação de novos artigos, bem como a sua modificação. Sendo este um módulo referente a manutenção, também é possível correr o agregador a pedido.

2.2 Servidor de Vendas

O Servidor de vendas trata todos os pedidos dos Clientes de vendas, quer sejam vendas ou entradas de stock, ou informações sobre o preço e stocks existentes no momento.

2.3 Cliente de Vendas

O Cliente de Vendas é a interface que o utilizador final usa para realizar todas as vendas e pedidos de informação, comunicando com o servidor de vendas.

2.4 Agregador

O Agregador tem como finalidade agregar vendas, ou seja, juntar todas as vendas referentes ao mesmo artigo de forma a poupar espaço.

Capítulo 3

Arquitetura e Solução do Projeto

3.1 Manutenção de Artigos

A Manutenção de Artigos trata de manter artigos, respetivos preços e nomes. Para manter essa informação usa dois ficheiros, *strings* e *artigos*. O utilizador consegue adicionar e modificar artigos através do stdin. Embora a manutenção de vendas seja independente dos restantes módulos, este comunica com o Servidor de Vendas, caso este esteja a correr, de forma a manter a sincronização da informação sobre o preço e numero de artigos existentes em ambos os lados. Para manter o tamanho do ficheiro de *strings* o mais pequeno possível, foi implementado um compactador para esse ficheiro, garantindo assim que não existem grandes volumes de dados desatualizados.
Aqui também é possível correr o agregador concorrente sobre o ficheiro de vendas.

3.2 Servidor de Vendas

O Servidor de Vendas trata dos pedidos de todos os clientes, e trata de manter informação sobre o stock de cada artigo. Para manter essa informação utiliza o ficheiro *stocks*. O Servidor corre em background, consistindo em três processos distintos que tratam de tarefas diferentes:

- Sincronização com a Manutenção de Artigos (preços e criação de novos artigos), com a finalidade de manter, tanto o ficheiro de stocks, como a cache de preços sempre atualizada.
- Conter a cache de preços.
- Comunicação e tratamento dos pedidos dos clientes.

Toda a comunicação com o servidor é feita com recurso a *Named pipes*, existindo um pipe para os clientes efetuarem pedidos e um para o Manutenção de Artigos enviar as alterações que ocorram da parte dele.

3.3 Cliente de Vendas

O Cliente de Vendas interage com o utilizador a partir do *stdin* e *stdout*, e interage com o Servidor de Vendas enviando informação para o pipe de entrada que o Servidor cria, e recebe a resposta deste através de um pipe que ele cria, tendo como identificador o seu *PID*, de forma a garantir que as informações não são lidas por outro cliente. Para isto acontecer, para além do pedido, é escrito no pipe de leitura do servidor o *PID* do cliente.

3.4 Agregador

O agregador recebe input do *stdin* linhas com o formato estipulado para o ficheiro *vendas*, calculando os totais de cada artigo, devolvendo esses totais no mesmo formato pelo *stdout*.

Capítulo 4

Cache de Preços

A cache de preços, parte integrante do Servidor de Vendas, existe para evitar consultar o ficheiro de Artigos a cada pedido do cliente. Para a manter sempre sincronizada com a informação que existe na parte da Manutenção de artigos, optamos por implementar esta num processo distinto do Servidor, afim de não causar atrasos nas respostas aos pedidos dos clientes. Assim, quando o servidor necessita da informação sobre um preço de um artigo, envia o pedido através do pipe criado para o efeito.

Capítulo 5

Agregador Concorrente

O Agregador Concorrente, implementado na Manutenção de Artigos, consiste em correr o agregador sobre o ficheiro vendas, dividindo o em partes, sendo capaz de agregar essas pequenas partes em paralelo e por fim agregar as partes mais pequenas todas, tornando assim a agregação deste ficheiro muito mais rápida.

Capítulo 6

Sincronização Artigos Stock

Como a informação contida no ficheiro Stocks tem que estar sempre atualizada com o ficheiro Artigos, ao ser criado o ficheiro Artigos, é guardada a informação de quando foi criado. Esta mesma marca temporária é transcrita para o ficheiro Stocks na sua criação. Para assegurar que o ficheiro Stocks tem a informação referente ao ficheiro Artigos mais atualizado, caso essa marca difira entre ficheiros, o ficheiro Stocks é totalmente refeito. Também quando são adicionados novos artigos, essas alterações são comunicadas ao Servidor, que trata de actualizar o ficheiro. Quando o servidor não esta a correr, essa verificação e feita ao arranque do mesmo, tendo assim a certeza que as referencias a artigos do ficheiro Stocks são corretas e as mais recentes.

Capítulo 7

Testes

7.1 Manutenção de Artigos

Para testar a Manutenção de Artigos foram criados dois programas em Python. O primeiro gera input em que as instruções dadas são válidas mas aleatórias. O segundo gera um conjunto de instruções que é válido e é possível testar se estão a ser bem interpretadas.

- Adiciona N produtos (passados como argumento), em que o nome é igual ao id do produto e o preço é igual a 0.
- Altera o nome de todos os produtos para 10 vezes o seu id.
- Altera o preço de todos os produtos para ser igual ao seu id.

Assim, para ver se o resultado é o esperado basta verificar se para um dado id este tem preço igual a id e tem um nome igual a 10 vezes o id.

7.2 Cliente de Vendas

Para testar o Cliente de Vendas foram criados dois programas em Python que seguem a mesma lógica dos criados para a Manutenção de Artigos. O primeiro gera input em que as instruções dadas são válidas mas aleatórias. O segundo gera um conjunto de instruções que é válido e é possível testar se estão a ser bem interpretadas.

- Acede a todos os ids.
- Adiciona ao stock 100 artigos de cada artigo.
- Remove do stock 10 artigos de cada artigo.

Assim, para verificar se o resultado é o esperado podemos confirmar se para um dado id o stock disponível é igual a 90. Na mesma veia, se vários Clientes de Vendas correrem em concorrência, para validar os resultados é possível verificar se a fórmula $stock\ /\ 90 = N$, sendo o stock o stock disponível para um dado artigo e N o número de clientes de vendas que foram corridos em concorrência, é verdade.

Capítulo 8

Conclusão

Para concluir, conseguimos cumprir todos os requisitos propostos criando no processo um sistema de gestão de stocks capaz de utilizar uma arquitetura *Cliente, Servidor* que suporta várias centenas de clientes. Como trabalho futuro, gostaríamos de melhorar a relação entre o Manutenção de Artigos e o Cliente de Vendas.