Universidade do Minho

Mestrado Integrado em Engenharia Informática

Sistemas Operativos

Grupo 15:

Goncalo Esteves a85731
Ricardo Costa a85851
Rui Oliveira a83610

1 Introdução

No âmbito da Unidade Curricular de Sistemas Operativos, foi-nos proposta a elaboração de um projeto em linguagem de programação C, tendo por base o uso de *System Calls*, de modo a criar um protótipo de um sistema de gestão de inventário e vendas. Este sistema seria constituído por vários programas: manutenção de artigos, servidor de vendas, cliente de vendas e agregador de dados. O objetivo do trabalho será desenvolver um *software* capaz de guardar diversos dados em ficheiros, utilizando as técnicas lecionadas nas aulas.

2 Programas

2.1 Manutenção de Artigos

Para este programa, foi-nos pedido que ele fosse capaz de permitir a inserção de novos artigos, e a alteração de atributos de artigos previamente inseridos. Tendo cada artigo um código numérico associado, o programa seria capaz de alterar os atributos de um dado artigo, sabendo o seu código e o seu novo atributo.

Deste modo, o nosso programa cria os ficheiros **artigos** e **strings**, e abre os pipes com nome **artigosAux** e **agregadorAux** com permissões de leitura, que servirão para comunicar com o Servidor de Vendas.

Tal como foi pedido, o nosso programa lê todo o seu input pelo stdin, filtrando apenas as inserções válidas. De tal modo, ele verifica se a primeira letra inserida, que é também a primeira "palavra" lida do stdin, é um "i", um "n", um "p" ou um "a". Para todos os restantes casos, o programa apenas regista num ficheiro de erros que houve um pedido de operação inválido.

No caso de a primeira letra ser um "i", ele fornece à função *insereArtigo* todo o input lido, já distribuído por um array com 3 *char**; aquele que será o código do produto; e a referência onde será guardado o nome do produto no ficheiro **strings**. Esta função atualiza os ficheiros de **artigos** e **strings**, através do uso de um *fork*: enquanto que o "filho" trata da inserção no ficheiro *artigos* do código do produto, da referência onde está guardado o nome do artigo e do preço do artigo, o "pai" trata da inserção no ficheiro **strings** do nome do produto. Para além disto, o programa envia ao Servidor de Vendas o código do artigo, através do pipe com nome **artigosAux**, de modo a que este atualize o ficheiro de **stocks** com o novo artigo.

Quando a primeira letra é um "n", o programa verifica, antes de tudo, se o código de produto ao qual se pretende alterar o número é válido (ou seja, se o código de produto é inferior ou igual ao número de linhas do ficheiro **artigos**, uma vez que cada linha corresponde a um único produto). Caso este seja, o programa cria um fork, onde o "filho" recorre a função *insereNome*, passando-lhe o novo nome do produto, e o "pai" atualiza a referência do nome do produto no ficheiro **artigos** através do uso da função *atualizaNome*, passando-lhe o código do produto a alterar e a nova referência. A função *insereNome* insere no ficheiro

strings o nome passado como argumento, na próxima referência. A função *atualizaNome* vai à posição onde está guardado o artigo com o código de produto dado como argumento, e altera a referência para a nova referência, passada como argumento.

Se a primeira letra é um "p", então o programa irá antes de mais verificar se o código de produto dado é válido. Em caso afirmativo, o programa irá invocar a função *atualizaPreco*, à qual é passada o código do artigo ao qual se quer alterar o preço, e o novo preço. Esta função percorre o ficheiro **artigos** até à posição do código, e atualiza o seu preço.

Caso a primeira letra seja um "a", o nosso programa irá comunicar com o Servidor de Vendas através do pipe com nome **agregadorAux** requerindo-lhe que este agregue todas as vendas efetuadas desde a última agregação.

2.2 Servidor de Vendas

Para este programa, foi-nos pedido que ele fosse capaz de controlar stocks, receber pedidos do cliente de vendas, e registar as vendas efectuadas. Além disso, o servidor de vendas providencia ainda para correr o agregador a pedido, fazendo com que este receba o intervalo (para ser agregado) do ficheiro de vendas desde a última agregação.

O programa começa por criar os ficheiros **stocks** e **vendas**. Seguidamente, de forma iterada, criamos processos "filho" com recurso ao fork(). O primeiro filho cria o pipe com nome **artigosAux**, abre-o com permissões de leitura e lê o seu conteúdo, escrevendo-o no ficheiro de stocks.

O segundo filho cria o pipe com nome **agregadorAux**, abre-o com permissões de leitura e executa o agregador.

O terceiro filho cria o pipe com nome **clienteMostrarsAux**, abre-o com permissões de leitura e mostra no stdout o stock e preço de determinado artigo recorrendo a uma função auxiliar denominada *mostrarStock*. Esta função abre o ficheiro de stocks, vai até a posição onde está guardado o artigo com o código de produto passado como argumento e imprime a quantidade do mesmo em stock, bem como o seu preço unitário.

Por último, o quarto filho cria o pipe com nome **clienteAlterarAux**, abre-o com permissões de leitura e altera o stock de determinado artigo usando uma função chamada *alterarStock*. Esta função auxiliar vai à posição onde está guardado o artigo com o código de produto dado como argumento e reescreve a sua quantidade com o inteiro passado como argumento. Caso o inteiro seja negativo, tratasse de uma venda, logo usando outra função auxiliar, *alterarVendas*, abrimos o fihceiro de vendas e escrevemos o código de produto que estamos a tratar, a quantidade vendida e o montante total desta venda.

2.3 Cliente de Vendas

Para este programa, foi-nos pedido que ele fosse capaz de comunicar com o servidor de vendas, sendo capaz de solicitar-lhe a execução de dois tipos distintos de operações: a primeira, que é distinguível uma vez que apenas introduz um

parâmetro (um código númerico) é utilizada para requesitar a mostragem da quantidade em stock e do preço de um dado produto (indicado pelo código); a outra, que introduz dois parâmetros diferentes, é utilizada para efetuar vendas ou entrada de stock, introduzindo o código do produto e a quantidade (negativa ou positiva).

O programa Cliente de Vendas cria os ficheiros **stocks** e **vendas**, e abre os pipes com nome **clienteMostrarAux** e **clienteAlterarAux** com permissões de escrita, por forma a comunicar com o Servidor de Vendas. Para além destes, é criado um terceiro pipe cujo nome será o pid do processo, por onde serão recebidas as respostas do Servidor de Vendas.

Neste programa, apenas é aceite que se insira, no máximo, dois parâmetros pelo stdin. Quando são inseridos mais do que 2 parâmetros, o programa regista num ficheiro de erros que houve um excesso de argumentos recebidos.

Quando apenas é passado um parâmetro, este terá de ser um código e, como tal, escreve-se no pipe **clienteMostrarAux** este código e o pid do processo, ficando depois o programa à espera que seja inserido no pipe cujo nome é o seu pid, a resposta do servidor.

Quando são passados dois parâmetros, o programa envia ao Servidor de Vendas o código numérico e a quantidade (os parâmetros recebidos) e o seu pid através do pipe **clienteAlteraAux**, e aguarda pela resposta do servidor, que será envidada para o pipe cujo nome é o seu pid.

2.4 Agregador

O objetivo deste programa é agregar o ficheiro **vendas** guardando o resultado desta agregação num ficheiro novo, cujo nome é a data e a hora na qual foi pedida a agregação. Sempre que é pedido uma nova agregação é criado um novo ficheiro, no qual aparece apenas uma única vez cada código de artigo, com a respetiva quantidade total e o respetivo montante total de vendas desse artigo que se encontravam no ficheiro **vendas**.

Este programa vai percorrendo o ficheiro **vendas**, linha a linha, e para cada código de artigo verifica primeiro se esse código já existe no ficheiro criado com a data e hora de quando o programa foi executado (recorrendo à função *procuraCodigoExistente*), pois se já existir, signifca que esse artigo já foi agregado. Se o código de artigo não existir no ficheiro da agregação, é chamada a função *procuraCodigoIgual* que vai percorrer o resto do ficheiro **vendas** e sempre que encontrar um código de artigo igual vai somando a quantidade e o montante. Depois de executada a função, escreve no ficheiro de agregação o resultado da agregação deste artigo e passa para a próxima linha do ficheiro **vendas** repetindo o processo.

O programa pode receber também um argumento que indica a linha do ficheiro **vendas** a partir da qual o utilizador quer fazer a agregação.

Para conseguirmos obter a data e hora de execução do programa é usada a biblioteca *time.h*, guardando esta informação numa *string*, que depois é usada para criar o ficheiro de agregação com o nome sendo a *string*.

3 Conclusão

Após a elaboração deste projeto, podemos afirmar que aprimoramos o nosso domínio sobre a linguagem de programação C, nomeadamente o uso das *System Calls*, que foram a base do nosso trabalho. Com o desenvolvimento deste trabalho, fomo-nos deparando com diversos obstáculos que se colocaram à nossa frente, sendo necessário recorrer a diversas técnicas aprendidas nas aulas, sabendo quando e como utiliza-las e aplicando-as da forma que tornariam o trabalho mais eficiente.