

UNIVERSIDADE DO MINHO

LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Sistemas Operativos

Gestão de Vendas

Autores:

Ilda Pinto	A78195
Sandro Cruz	A78708
Sónia Costa	A71506

13 de Maio de 2019

Projeto de grupo submetido no âmbito da unidade curricular de Sistemas Operativos do 2º ano da Licenciatura em Ciências da Computação da Universidade do Minho.

11 de Maio de 2019

Conteúdo

1	Introdução	4
2	Breve descrição do trabalho	5
2.1	Contextualização	5
2.2	Arquitectura do trabalho	6
3	Funcionalidades do sistema	7
3.1	Manutenção de artigos	7
3.2	Cliente de Vendas e Servidor de Vendas	9
3.3	Agregador	12
4	Análise dos programas	13
4.1	Manutenção de artigos	13
4.2	Comunicação entre Servidor de Vendas e Cliente	13
4.3	Servidor de Vendas	14
4.4	Cliente de Vendas	14
4.5	Agregador	14
4.6	Compactação do ficheiro Strings	15
4.7	Auxiliar.h	15
4.8	Auxiliar.c	15
4.9	Makefile	16
5	Conclusão	17

Lista de Figuras

1	Estruturação do sistema	6
2	Inserção de um novo artigo.	7
3	Alterar o nome do artigo.	8
4	Alterar o preço do artigo.	8
5	Inserção de um artigo no ficheiro stock.	9
6	Visão geral das funcionalidades do Cliente de Vendas e Servidor de Vendas.	9
7	Visualizar o preço e quantidade em stock.	10
8	Adicionar stock.	11
9	Efetuar uma venda e atualizar stock.	11
10	Desprezar entradas incorretas.	11
11	Resultado da agregação	12
12	Comunicação entre Servidor de Vendas e Cliente	13

1 Introdução

Este trabalho visa a construção de um protótipo de um sistema de gestão de inventário e vendas, constituído por vários programas: manutenção de artigos, servidor de vendas, cliente de vendas, e agregador de dados.

Já referente às funcionalidades avançadas foi desenvolvido o agregador concorrente de forma a tirar partido da eficiência do processador ao ser possível fazer o processamento de dados com vários processos e assim aumentar a eficiência deste programa. Por sua vez, o programa que realiza a compactação do ficheiro "*strings*" serve para reduzir o espaço utilizado em memória.

Ao longo do relatório serão explicados com mais precisão os programas que constituem o sistema e passos seguidos na construção do trabalho, assim como uma breve explicação dos mesmos. Serão ainda mostrados alguns dos testes realizados de forma a garantir um maior grau de fidelidade ao trabalho.

2 Breve descrição do trabalho

2.1 Contextualização

O objetivo do trabalho reside na construção de um sistema de gestão de inventário e vendas, para tal recorremos a quatro programas distintos:

O manutenção de artigos (ma) que possibilita a inserção de novos artigos (especificando o nome e preço de venda), ou alteração do nome ou preço de um dado artigo. Cada artigo tem um código numérico, atribuído na criação como o próximo de uma sequência.

O servidor de vendas(sv) controla stocks, recebe pedidos do cliente de vendas(cv), e regista as vendas efectuadas (acrescentando uma entrada a um ficheiro VENDAS, contendo código, quantidade e montante total da venda). Para além disso, o servidor de vendas corre em paralelo o agregador, fazendo com que este receba o intervalo (para ser agregado) do ficheiro de vendas desde a última agregação e escrevendo o resultado da agregação num ficheiro cujo nome reflete o momento em que a agregação foi solicitada.

O cliente de vendas(cv) interage com o servidor de vendas, solicitando-lhe a execução de operações que se distinguem pelo número de parâmetros introduzidos. Uma das operações retorna a quantidade em stock e o preço de um artigo (identificado pelo código). A outra operação permite efetuar vendas ou entrada em stock, especificando o código e quantidade (negativa ou positiva, respetivamente). O sistema permite a execução concorrente de vários clientes de vendas.

O programa agregador(ag) funciona como filtro. Recebe pelo stdin entradas no formato do ficheiro de vendas, até end-of-file, produzindo para o stdout os dados agregados de cada artigo com vendas efectuadas, contendo o código do artigo, a quantidade total e o montante total de vendas do artigo respectivo (mantendo o formato do ficheiro de vendas).

O programa de compactação do ficheiro *strings* (cmp) foi desenvolvido com o intuito de reduzir o desperdício de espaço. Quando este superar os 20% do ficheiro, é feita a compactação no mesmo ficheiro e consequentemente ajustadas as posições no ficheiro de artigos.

2.2 Arquitectura do trabalho

O esquema seguinte representa a arquitetura do sistema de gestão de inventário e vendas, por nós desenvolvido:

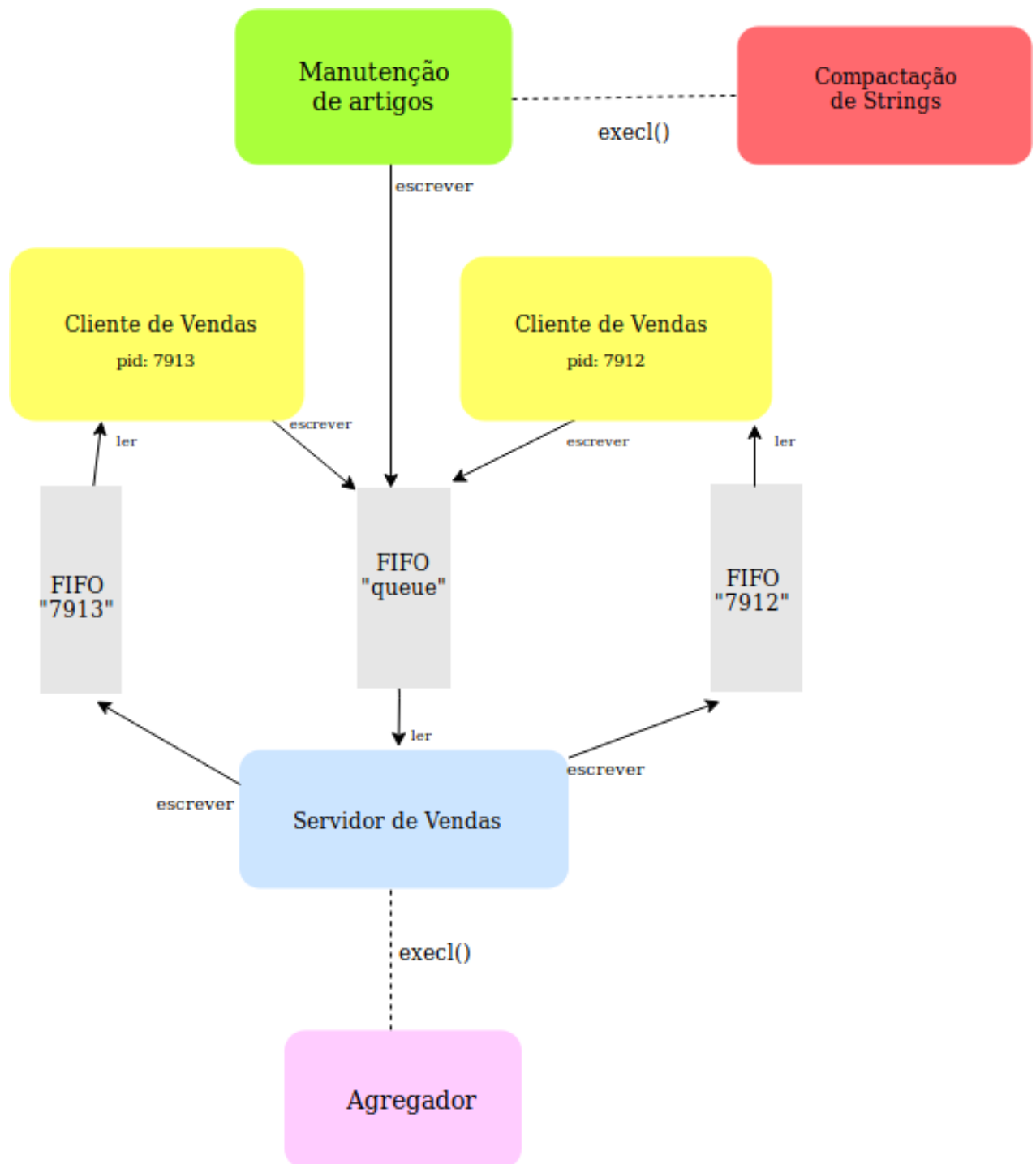


Figura 1: Estruturação do sistema

3 Funcionalidades do sistema

Nesta secção são facultados *prints* de exemplo das funcionalidades fornecidas pelo sistema.

3.1 Manutenção de artigos

No programa Manutenção de Artigos, é possível inserir um artigo, alterar o nome e o preço. A informação é guardada em dois ficheiros distintos *artigos* e *strings*.

```
artigos.txt x ... strings.txt x
0      1      0      | 1 melancia
1      4      9      | 2 manteiga de amendoim
2      2      30     | 3 bolachas de chocolate
3      5      52     | 4 aveia com sabor a morango
4      3      78     | 5 mousse de chocolate
                        | 6

TPUT  DEBUG CONSOLE  TERMINAL  4: sv, ma, bash
'::~/Desktop/ ilda@grupo17:~/Desktop/sotrabalho$ ./ma
./sv
i melancia 1
0
i manteiga de amendoim 4
1
i bolachas de chocolate 2
2
i aveia com sabor a morango 5
3
i mousse de chocolate 3
4
```

Figura 2: Inserção de um novo artigo.

id	nome	preco	quantidade
1	melancia	0	1
2	melão	9	2
3	morangos	40	1
4	geleia	32	4
5			

```

strings.txt
1 melancia
2 melão
3 morangos
4 geleia
5 geleias
6 morango
7

Terminal
ilda@grupo17:~/Desktop/sotrabalho$ ./ma
n 3 geleias
n 2 morango
_

```

Figura 3: Alterar o nome do artigo.

id	nome	preco	quantidade
1	melancia	0	1
2	melão	9	5
3	morangos	40	3
4	geleia	32	4
5			

```

strings.txt
1 melancia
2 melão
3 morangos
4 geleia
5 geleias
6 morango
7

Terminal
ilda@grupo17:~/Desktop/sotrabalho$ ./sv
p 1 5
p 2 3
_

```

Figura 4: Alterar o preço do artigo.

Para além disso, quando é inserido um novo artigo no programa Manutenção de Artigos, este envia uma mensagem para o Servidor de Vendas através do *fifo "queue"* com o identificador do artigo, que por sua vez insere este artigo no ficheiro *stocks* com quantidade zero.

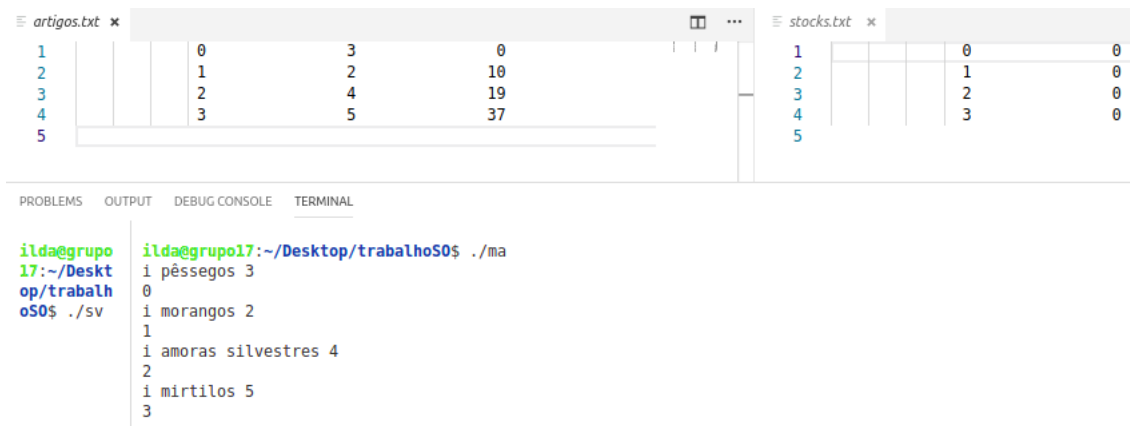


Figura 5: Inserção de um artigo no ficheiro stock.

3.2 Cliente de Vendas e Servidor de Vendas

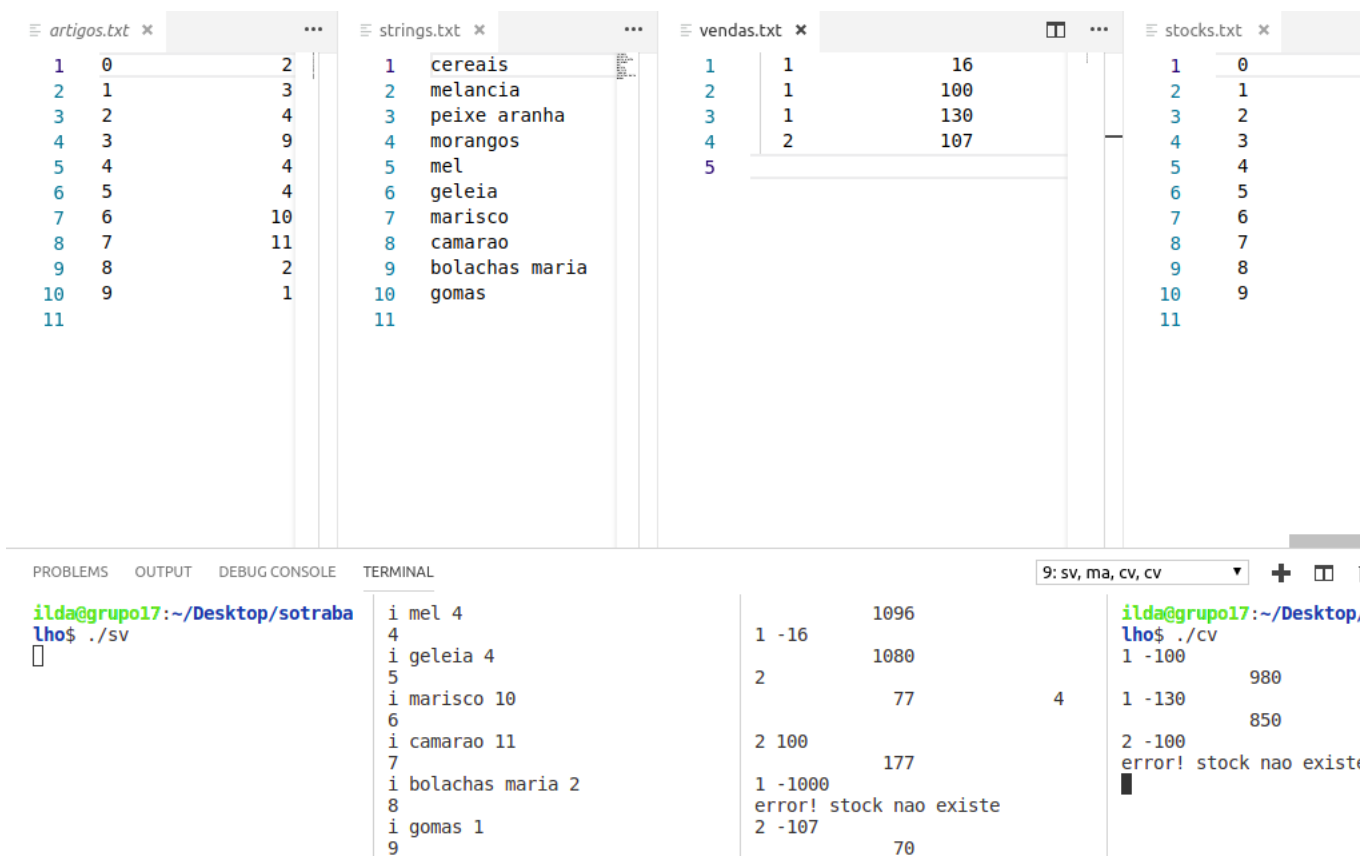


Figura 6: Visão geral das funcionalidades do Cliente de Vendas e Servidor de Vendas.

O programa Cliente de Vendas proporciona diferentes funcionalidades ao utilizador deste Sistema.

- a) Permite a execução concorrente de vários clientes de vendas, todavia não é possível registrar essa funcionalidade através de um *print*.
- b) Quando inserimos um identificador de um artigo no *standard input* é retornado para o *standard output* a quantidade desse artigo em stock e o preço:

The screenshot shows a code editor with three files open: `artigos.txt`, `strings.txt`, and `stocks.txt`. Below the editor is a terminal window showing the execution of a program.

artigos.txt

1	0	1
2	1	4
3	2	2
4	3	5
5	4	3
6		

strings.txt

1	melancia
2	manteiga de amendoim
3	bolachas de chocolate
4	aveia com sabor a moran
5	mousse de chocolate
6	

stocks.txt

1	0	83
2	1	86
3	2	77
4	3	15
5	4	93
6		

Terminal

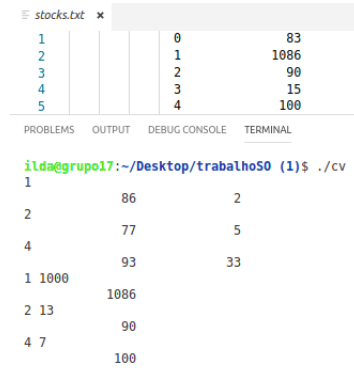
```

ilda@grupo17:~/Desktop/sotrabalho$ ./cv
4
3
2
_
93
15
77
3
5
2

```

Figura 7: Visualizar o preço e quantidade em stock.

c) Quando inserimos um identificador e um número positivo no *standard input*, o *stock* referente a esse artigo é atualizado no ficheiro *Stock* e é retornado para o *standard output* a quantidade em *stock* desse artigo.

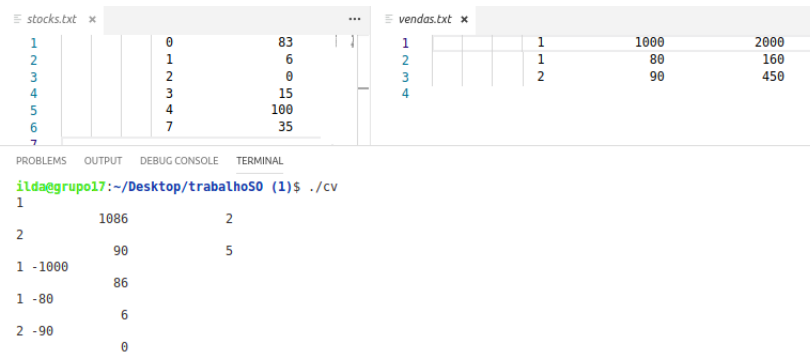


stocks.txt		
1	0	83
2	1	1086
3	2	90
4	3	15
5	4	100

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
1	86	2	
2	77	5	
4	93	33	
1 1000			
2 13	1086		
4 7	90		
	100		

Figura 8: Adicionar stock.

d) Quando inserimos no *standard input* um identificador e um número negativo, caso exista quantidade em *stock*, é efetuada uma venda, caso contrário a venda não será efetuada. O artigo é atualizado no ficheiro *Stock* e é registada uma venda no ficheiro *Vendas*. Por fim, é retornado para o *standard output* a quantidade desse artigo em *stock*.



stocks.txt		
1	0	83
2	1	6
3	2	0
4	3	15
5	4	100
6	7	35
7		

vendas.txt			
1	1	1000	2000
2	1	80	160
3	2	90	450
4			

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
1	1086	2	
2	90	5	
1 -1000	86		
1 -80	6		
2 -90	0		

Figura 9: Efetuar uma venda e atualizar stock.

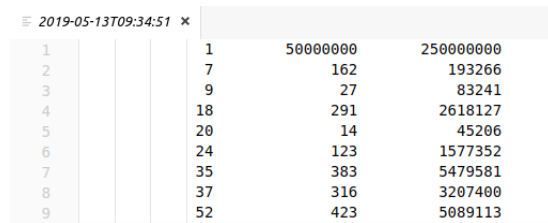
e) Para além disso, tivemos atenção ao tratamento do *input*, para que sejam desprezadas entradas que não sejam números

```
ilda@grupo17:~/Desktop/trabalho50 (1)$ ./cv
1ssds
1 100adad
1 ss
2sd -110
90
error! id nao existe
```

Figura 10: Desprezar entradas incorretas.

3.3 Agregador

O agregador pode ser executado através do terminal ou pode ser chama através de um pedido ao programa ma, através do comando "a". O agregador não tem parametros de entrada pois trata a informação diretamente a partir do ficheiro *Vendas*. Este programa usa um ficheiro auxiliar chamado *agregs.txt* que guarda o último bit lido na execução anterior, para que não se agregue multiplas vezes a mesma venda.



1	1	50000000	25000000
2	7	162	193266
3	9	27	83241
4	18	291	2618127
5	20	14	45206
6	24	123	1577352
7	35	383	5479581
8	37	316	3207400
9	52	423	5089113

Figura 11: Resultado da agregação

4 Análise dos programas

Neste capítulo, é feita uma explicação das decisões tomadas e uma análise cuidada dos programas de forma a resumir o funcionamento do trabalho prático.

4.1 Manutenção de artigos

O programa pode receber quatro opções distintas 'i', 'n', 'p' ou 'a' que significam inserção, alteração do nome, alteração do preço e agregação, respetivamente. Ao programa manutenção de artigos estão associados dois ficheiros, o ficheiro *artigos* e o ficheiro *strings*.

No ficheiro *artigos* cada linha é composta por três campos distintos, o código identificador do artigo, o preço e a referência ao nome no ficheiro *strings*. Cada campo é separado por um espaço, tendo um tamanho fixo de quinze caracteres e um *newline* no fim. Optamos por definir um tamanho fixo para cada campo, com o intuito de evitar erros e facilitar o tratamento de dados. A referência ao nome é a posição onde este se encontra no ficheiro *strings*, sendo esta calculada através do número de caracteres desde o início do ficheiro até à posição onde este foi escrito. Por sua vez, o ficheiro *strings* é constituído por nomes de tamanho variável. Recorremos à contagem das linhas dos ficheiros *artigos* e *strings*, para que os dados persistam caso o servidor seja fechado. A alteração de nome e preço é feita através do cálculo exato da posição deste valor na linha correspondente ao id do artigo, e recorrendo à função *lseek* para posicionar o *offset* do ficheiro.

4.2 Comunicação entre Servidor de Vendas e Cliente

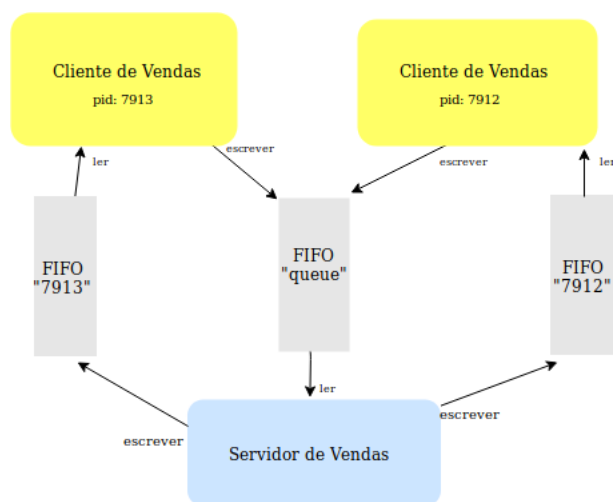


Figura 12: Comunicação entre Servidor de Vendas e Cliente

A comunicação entre o servidor e o cliente é estabelecida através de *fifos*. Existe um *fifo* principal que funciona como uma *queue* para receber mensagens de todas as instâncias do Cliente de Vendas que por sua vez serão posteriormente lidas pelo servidor. O conteúdo de cada mensagem é dependente da funcionalidade pedida, porém o primeiro campo é sempre o nome de um *fifo* associado

à instância de Cliente de Vendas que enviou a mensagem. O nome do *fifo* enviado é determinado através do pid do processo cliente, de forma a garantir a unicidade do nome.

Optamos por esta estratégia, pois várias instâncias de cliente de vendas precisam de trocar informação com o servidor e este terá de escrever mensagens diferentes para cada instância. Se existisse um único *fifo* iria ocorrer troca de dados e instâncias iriam obter respostas que não lhes correspondem.

4.3 Servidor de Vendas

O servidor de vendas é um programa que faz a gestão de todos os pedidos provenientes dos programas "Manutenção de artigos" e "Cliente de Vendas".

Os pedidos chegam ao servidor através de um *fifo*, este analisa o pedido através do número de argumentos recebidos. É possível receber um, dois ou três argumentos.

O "Manutenção de artigos" envia ao servidor de vendas um único argumento, que por sua vez é o identificador de artigo. O servidor abre o ficheiro "*stocks*", onde cria uma linha para esse identificador com a quantidade de stock igual a zero.

Por sua vez, o "Cliente de Vendas" pode enviar ao servidor mensagens com dois ou três argumentos. Quando envia uma mensagem com dois argumentos, estes são o nome do *fifo* associado ao cliente que enviou o pedido e o identificador do artigo para consulta. Quando o "Cliente de Vendas" envia ao Servidor três argumentos estes são o nome do *fifo*, o identificador do artigo e a quantidade. Caso a quantidade seja positiva o servidor abre o ficheiro "*stocks*" e acrescenta essa quantidade ao stock do artigo respetivo. Caso a quantidade seja negativa, se essa quantidade existir em stock é realizada uma venda que será registada no ficheiro "*vendas*" e o stock é atualizado, caso a quantidade em stock não seja suficiente para satisfazer o pedido, nada acontece. Cada venda guardada no ficheiro de vendas tem um tamanho fixo por linha, sendo 15 caracteres para o código, 15 para o montante de produtos vendidos, 15 para o montante total do preço, separados por um espaço e finalizando um *newline*, sendo um total de 48 caracteres por linha.

Para terminar a execução do servidor de vendas deve ser enviado um *ctrl+c*. Através do tratamento do sinal, o servidor esperará um momento em que nenhuma execução foi pedida para sair do programa corretamente.

4.4 Cliente de Vendas

O cliente de vendas lê o que foi inserido pelo utilizador do programa, verifica se o que foi inserido é válido, trata os dados e envia um pedido para o servidor.

4.5 Agregador

O agregador serve para agregar as vendas pelo código do produto. Este programa não recebe argumentos, acedendo diretamente à informação do ficheiro de vendas. Este programa possui um ficheiro auxiliar onde guarda o *offset* da última execução para que não se agregue duas vezes a mesma venda. Este

baseia-se na estrutura de cada venda para fazer a agregação. O agregador tem uma versão concorrente que pretende minimizar o tempo de cada agregação. Este envolve executar o processo filho por cada 10000 linhas de vendas e por fim fazer uma última agregação dos resultados.

4.6 Compactação do ficheiro Strings

Para fazer a compactação de *strings* foi necessário criar um novo programa "cmp", que por sua vez é invocado quando o desperdício de espaço superar os 20% do ficheiro "strings". Neste sentido é feito, no programa "Manutenção de artigos", o calculo que determina a razão entre o número de nomes alterados e o número de artigos inseridos

É criado um ficheiro auxiliar "stringsaux.txt" para onde são copiados os nomes atuais de cada artigo (recorrendo ao ficheiro "artigos" onde através do campo referência conseguimos saber qual o nome atual do artigo). Após isto, o ficheiro "strings" é truncado e o conteúdo do ficheiro "stringsaux" é copiado para o ficheiro strings. Por fim, a referência do nome no ficheiro "artigos" é atualizado para a referência atual e o ficheiro "stringsaux.txt" é truncado de forma a evitar erros numa próxima compactação.

4.7 Auxiliar.h

O ficheiro *Auxiliar.h* contém as assinaturas das funções que são necessárias para o uso dos programas anteriores e os *includes*. As funções estão definidas em *Auxiliar.c* que irá ser abordado posteriormente.

4.8 Auxiliar.c

Como foi dito, este ficheiro contém as funções que foram utilizadas no trabalho prático. Servem de complemento para a execução dos programas realizados. De forma detalhada, vai ser explicada cada função utilizada:

- `size_t gatherArg(char* arg[], char* buffer, size_t size);`

A função *gatherArg* recebe de argumentos um buffer (String) que possui tamanho *size* e no *arg* são armazenadas as palavras do buffer. Devolve o número de Strings que foram guardadas no *arg*.

- `size_t vectorToString(char* arg[], char* string, int init, int end);`

A função *vectorToString* é constituída por uma String *string*, um int *init*, um int *end* e um *arg* que é um array de Strings. Consiste na conversão de um array de Strings (vector) desde a posição *init* até à posição *end* para uma String. Retorna o comprimento dessa String.

- `ssize_t readln(int fildes, void *buf, size_t nbytes);`

Esta função lê uma linha a partir do *file descriptor* que é recebida como argumento na variável *fdes* até um total de *nbytes* guardando o que lê no array *buf*. Devolve o tamanho do *buf*. Devolve -1 em caso de erro.

- `int isNumber(char *string);`

A função *isNumber* recebe uma String, percorria, e verifica se cada caracter dessa String é um dígito. Caso todos os caracteres sejam um dígito (ou seja, um número) retorna o inteiro 1. Caso contrário, retorna o inteiro 0.

- `int isStock(char *string);`

Esta função é semelhante à anterior. No entanto, em vez de verificar somente se cada caracter dessa String é um dígito, verifica também se existe o caracter '-' para saber se esse número é negativo ou não. Retorna o inteiro 1 caso todos os caracteres verifiquem a condição, e o inteiro 0 caso contrário.

- `int escreverFifo(char* nome, char* frase);`

A função apresentada recebe como argumentos duas Strings. Consiste na escrita da String *frase* para o fifo com o nome apresentado na String *nome*, abrindo e fechando o respectivo fifo.

- `void formatTime(char* buffer);`

A função *formatTime* recebe uma String *buffer*. Guarda nesse *buffer* o formato do tempo que é pedido no enunciado do trabalho prático.

4.9 Makefile

A *makefile* possui 4 opções que são *compile*, *run* e *rm* que se descrevem a seguir:

- *rm*, que remove os arquivos que foram criados após a execução dos programas do trabalho prático de forma a não interferir com novas execuções.
- *compile*, que compila os ficheiros *ma.c*, *cv.c*, *sv.c*, *ag.c*, *cmp.c* e *ag-opti.c* juntamente com *lib/auxiliar.c* cada um e dá, respetivamente, o nome *ma*, *cv*, *ag*, *cmp* e *ag-opti* aos executáveis que foram criados.
- *run*, que executa os executáveis *ma* e *cv* de acordo com os exemplos colocados.

5 Conclusão

Neste trabalho prático, foram colocadas em prática algumas das funcionalidades que existem no sistema operativo. De forma generalizada, foram abordados vários temas essenciais tais como acesso a ficheiros, gestão de processos, execução de programas, redireccionamentos de descritores de ficheiros, pipes anónimos, pipes com nome e sinais. Colocaram-se em prática esses temas neste trabalho prático com sucesso nas funcionalidades básicas.

Por fim, foi possível obter uma compreensão reforçada da estrutura e principais funções do sistema operativo.