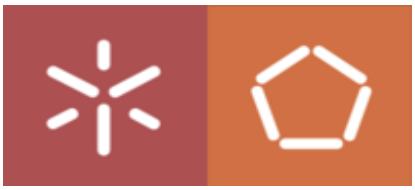
**Universidade Do Minho**



Mestrado Integrado em Engenharia Informática

# Relatório do Projeto de Sistemas Operativos

**Grupo 95:**

*Bruno Teixeira A84430*

*Diogo Rio A84752*

*Jorge Cerqueira A85573*

*2018/2019*

# Índice

Introdução ..... 3
Descrição do problema ..... 3
Implementação ..... 4
Manutenção de artigos ..... 4
Servidor e Cliente ..... 4
Agregação ..... 5
Ilustrações ..... 6
Conclusão ..... 7

# Introdução

Este trabalho foi proposto na unidade curricular de sistemas operativos sendo o seu objetivo a construção de um protótipo de um sistema de gestão de inventário e vendas.

Para a implementação deste projeto foram utilizados diversos conceitos lecionados na unidade curricular de sistemas operativos como pipes, sinais e diversas system calls.

# Descrição do problema

O projeto deverá ser composto pelos seguintes programas:

**Servidor de vendas**: gere o stock dos vários artigos, recebe pedidos de venda, utiliza o agregador de forma a compactar as vendas

**Manutenção de artigos**: gere os vários artigos contendo operações para fazer alterações de preço e nome dos mesmos, envia pedidos de agregação ao servidor e sinaliza-lhe possíveis alterações em artigos

**Cliente de vendas**: envia diversos pedidos ao servidor como reposição de stock e vendas

**Agregador**: funde vendas relacionadas ao mesmo artigo de maneira a reduzir o espaço ocupado por as mesmas de modo a facilitar a leitura de ficheiros por pessoas foi decido implementar um programa extra.

**Printer**: possibilita o utilizador imprimir os diversos ficheiros gerados pelos outros programas num formato legível de texto.

# Implementação

## Manutenção de artigos

Para a manutenção de artigos são usados dois ficheiros “ARTIGOS” e “STRINGS”, o primeiro armazena a seguinte estrutura:

```
typedef struct item_container {
iid_t id;
price_t price;
n_len_t name_len;
n_ref_t name_ref;
} item_container_t;
```

Desta maneira podemos facilmente encontrar o nome do artigo no ficheiro “STRINGS” através da sua referência (offset no ficheiro) e tamanho, o ficheiro “STRINGS” contêm também nos bytes iniciais informação sobre a quantidade “lixo” (bytes não utilizados) nesse ficheiro, esta informação é posteriormente utilizada para fazer a compactação quando a percentagem de lixo é elevada.

A compactação faz a leitura de todos os ficheiros para um novo ficheiro temporário, sendo que depois o copia para o original, de maneira a não estar constantemente a reescrever todo o ficheiro, a verificação só é feita após todas as operações no programa serem realizadas (i.e. o descritor do standard input é fechado).

## Servidor e Cliente

Todos os pedidos ao servidor são realizados com a seguinte estrutura:

```
typedef struct request {
    req_t rt;
    pid_t cli_pid;
    iid_t id;
    stock_t amount;
}* request;
```

Mesmo não sendo necessárias todas as variáveis contidas na estrutura, a uniformização do seu tamanho ajuda à receção e envio das mesmas uma vez que as leituras e escritas podem ser feitas facilmente de forma atómica.

Os possíveis tipos de pedido são:

```
typedef enum {
      reqt_show         // Pedido para enviar preço e stock de um artigo
    , reqt_sale         // Pedido para concretizar uma venda
    , reqt_stock        // Pedido de reposição de stock
    , reqt_update_cach  // Pedido para atualizar o preço de um artigo
    , reqt_reload_cach  // Pedido para reabrir o descritor dos artigos
    , reqt_aggregate    // Pedido para agregar as vendas atuais
} req_t;
```

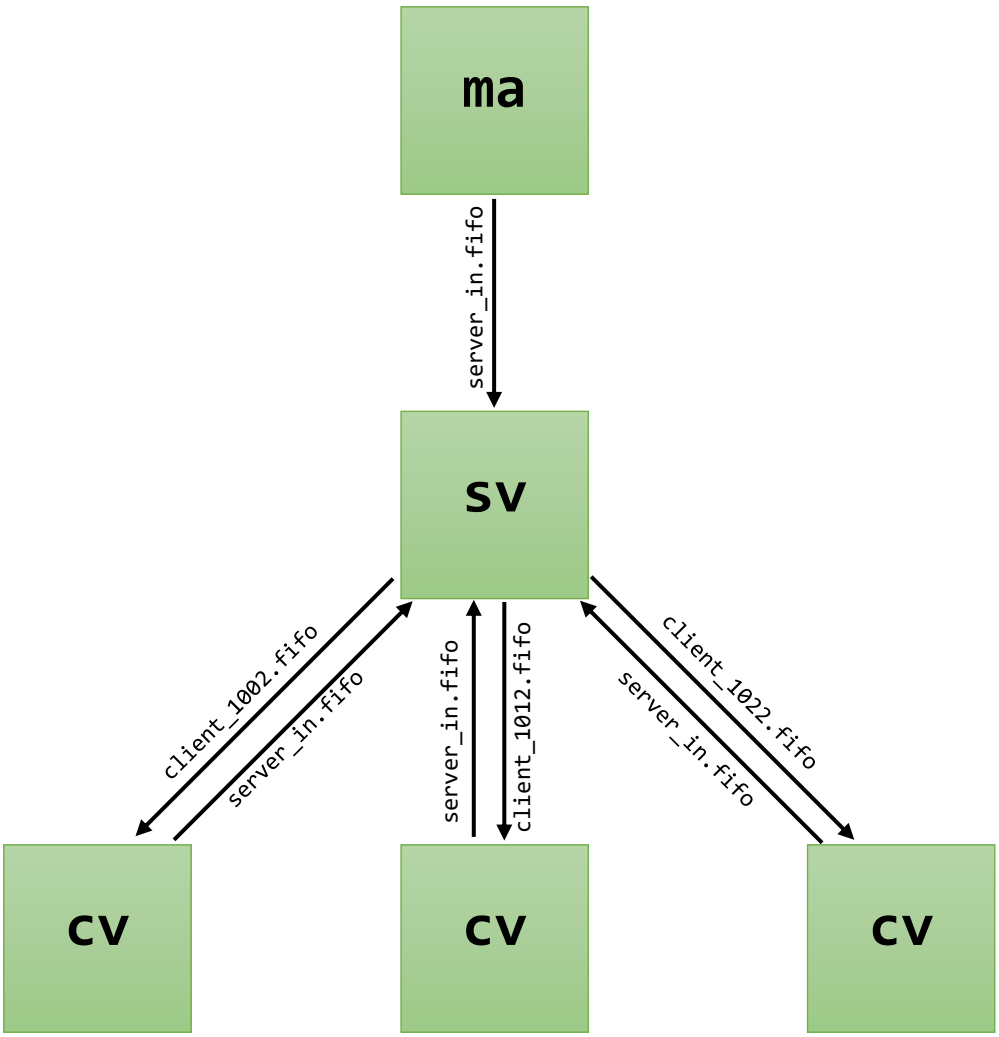
O cli_pid é dado ao servidor de modo a este poder saber para qual pipe enviar resposta ao pedido quando necessário, o pid do cliente é único no instante em que está aberto e por isso a sua pipe correspondente também o será (com formato "`client_$(pid).fifo`").

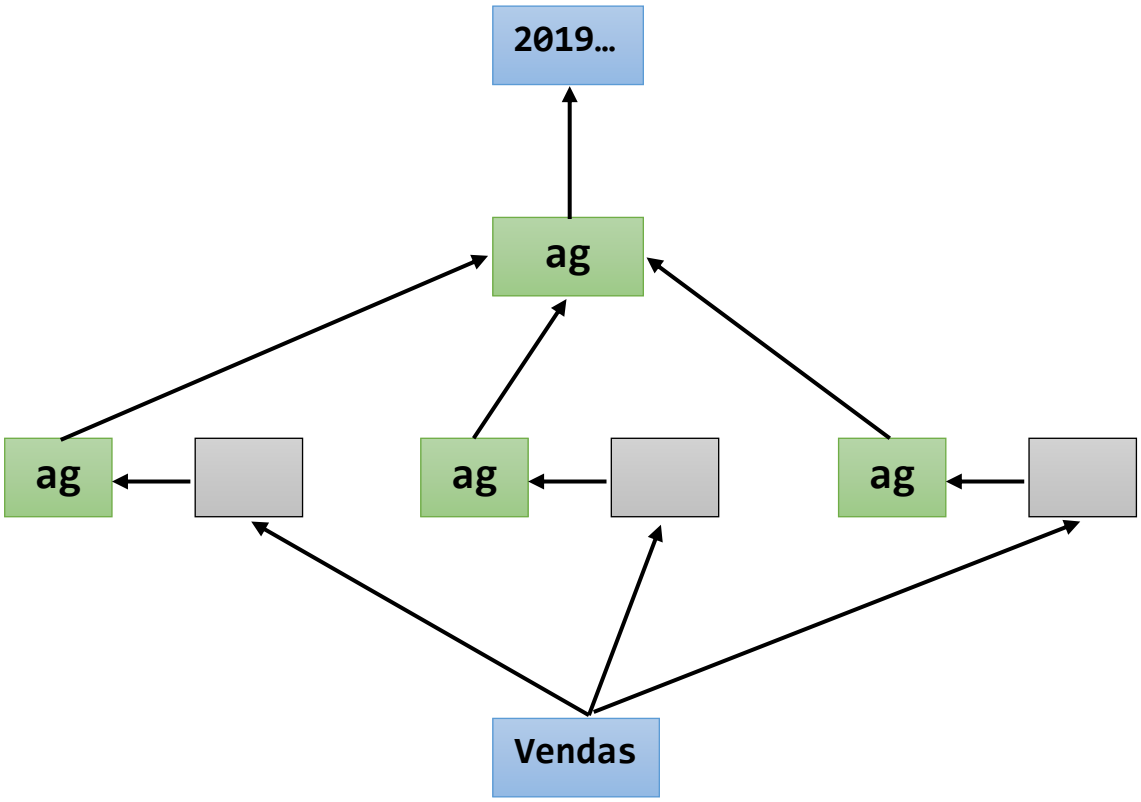
## Agregação

A agregação é feita de forma concorrente, o ficheiro de vendas é separado entre várias secções, para cada uma destas são criados dois processos, um deles é o agregador e o outro é faz a leitura das vendas do ficheiro e escreve-as para um descritor previamente conectado ao stdin do agregador.

O output de todos os agregadores é redirecionado para um outro agregador que junta as várias agregações feitas.

## Ilustrações






## Conclusão

Em suma, o grupo julga que os requisitos do trabalho foram alcançados.

Um dos aspetos a melhorar seria a implementação da cache. Sendo que é uma simples hash table em caso de colisão o valor é simplesmente alterado, no entanto, como algoritmia não é o foco da unidade curricular decidimos manter a estrutura da cache simples.