SISTEMAS OPERATIVOS

TRABALHO PRÁTICO | RELATÓRIO

META 2

INSTITUTO POLITÉCNICO DE COIMBRA
INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA
LICENCIATURA EM ENGENHARIA INFORMÁTICA

Henrique Barradas.: 201913583
João Carvalho.: 2019131769
TURMA P5

ANO LETIVO 2022/2023

Índice

Conteúdo

Índice de Figuras

1. Introdução

O presente trabalho prático tem como objetivo a implementação de um sistema de gestão de leilões.

O objetivo do sistema é funcionar como um fórum de mensagens. Este serve como intermediário entre clientes ao fazer, entre outros, a receção e entrega de mensagens entre estes. Assim, os clientes enviam mensagens que serão rececionadas pelo "backend", sendo depois entregues aos clientes consoante os comandos inseridos.

2. Estratégia Geral

Backend

A estratégia usada no processo *backend* foi usar um mecanismo *select* para lidar com a leitura de comandos através do teclado ao mesmo tempo que presta atenção aos FIFOS dos clientes e do heartbeat.

Sempre que for necessário comunicar com o cliente, o backend aproveita a estrutura *dataMSG* e é enviada através do FIFO do cliente respetivo.

Quanto à informação a ser guardada, criaram-se *arrays* dinâmicos para guardar todas as estruturas.

Quando o primeiro comando chega (login), passa pelo verificador e se estiver correto o backend manda uma mensagem de sucesso, de seguida o backend fica à espera de comandos vindos do frontend relativamente a gestão dos leilões.

O backend através de outro FIFO fica à espera de HEARTBEATS mandados por cada cliente (User) caso isto não aconteça o cliente é "kickado" do backend.

O backend pode ainda ser controlado por um "Administrador" que por si pode introduzir os comandos de admin.

Frontend

Quanto ao processo *Frontend*, é lá que o utilizador é criado e é onde se introduzem os comandos de utilizador. Para que o processo seja criado alem de dar valores às variáveis de ambiente (tanto pode ser no frontend como no backend) ,

É preciso correr o programa com os campos do login (./frontend nomeUser passUser).

Quanto à parte visual, foram criadas 2 janelas distintas. Uma que trata de mostrar o frontend onde se realizão os comandos e outra o backend que trata de mostrar certas mensagens e onde podem ser introduzidos os comandos admin.

3. Estrutura

3.1 Geral

A estrutura deste trabalho baseia-se num modelo servidor-cliente. O backend representando o servidor, está ligado e fica à espera de clientes que apareçam.Assim que a sua execução começar, há uma tentativa de acesso a um FIFO, impossibilitando a execução de 2 servidores em simultâneo caso esse FIFO já exista.

Para cada cliente será também criado um FIFO, de forma a que o servidor possa comunicar com os mesmos, sabendo exatamente a que cliente enviar a informação correta. Sempre que um cliente se ligar, desligar ou ser removido, o gestor é informado. Da mesma forma, se o gestor se desligar, todos os clientes ligados vão ser devidamente alertados.

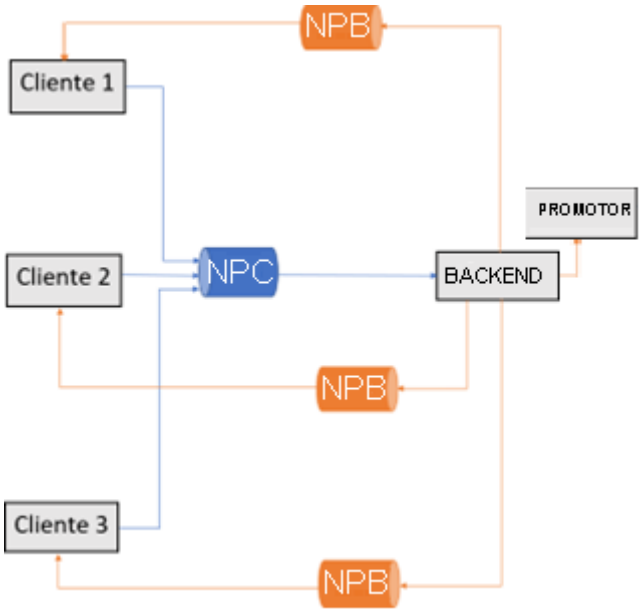

Figura 1 - Esquema da estrutura geral do sistema

Para resolver a problemática de comunicação backend promotores, foram implementados *pipes* anónimos apesar de essa função não estar implementada

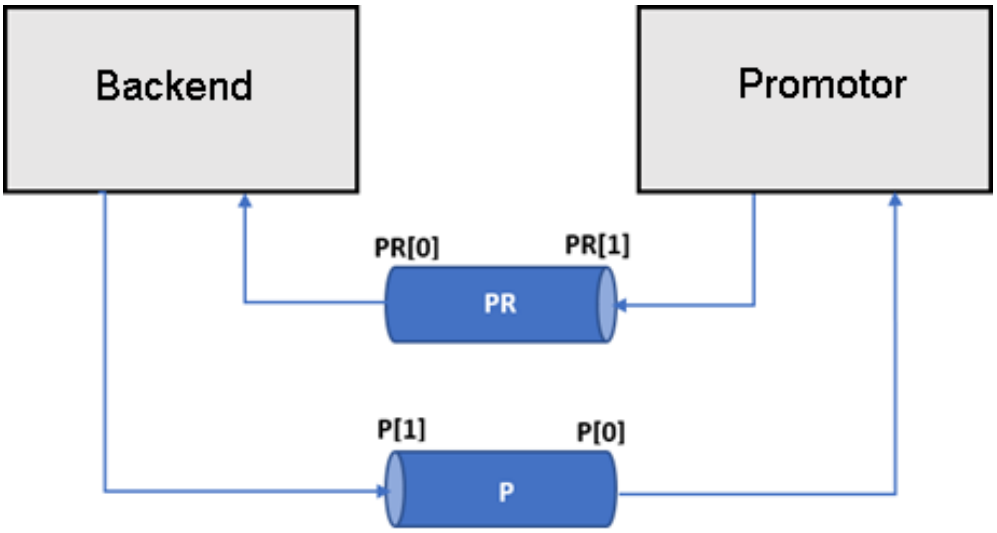

Figura 2 - Esquema da comunicação com o processo Promotor

3.2 Projeto

O projeto final contém as pastas backend_files,frontend_files,promotor_files e em comum temos um general.h que trata das estruturas etc. comuns.

.

4. Makefile

O ficheiro *Makefile* contém os *targets* de compilação "all", "backend", "frontend" , assim como "clean", que elimina todos os ficheiros temporários de apoioà compilação (ficheiros .o) e executáveis.

5. Ficheiros

Para este trabalho foram, então, criados os seguintes ficheiros.

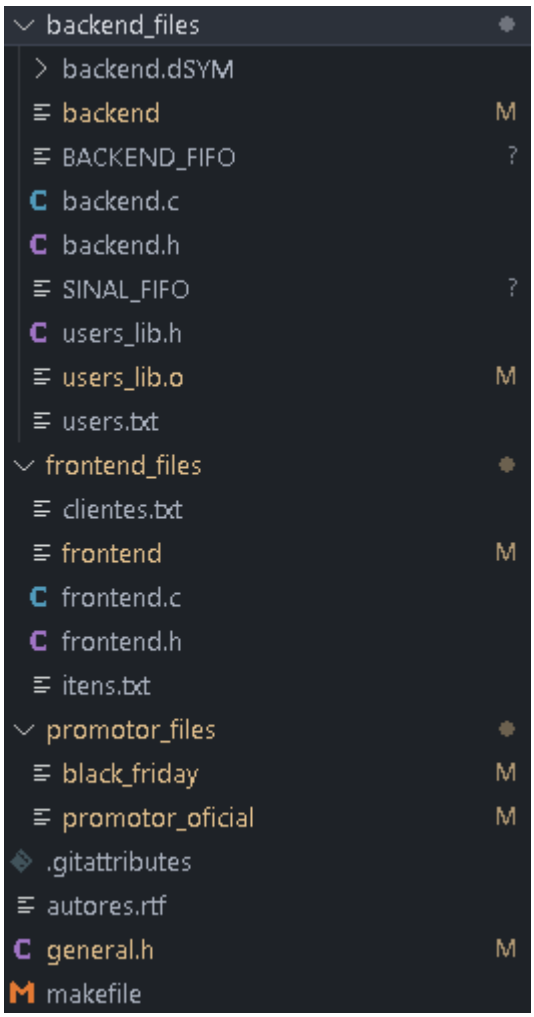

Figura 3 - Ficheiros

O ficheiro "frontend.c" recebe pela linha de comandos dois argumentos que correspondem ao username do cliente e a password. De seguida, backend verifica o login do cliente inserido e informa o cliente se foi logado ou não. Após essa operação, o cliente passa a ter disponível toda uma lista de comandos com as funcionalidades que lhe estão reservadas,podendo escolher o que pretende fazer.

O ficheiro de texto "users.txt" contém todas as informações de login guardadas e disponíveis.

O ficheiro "itens.txt" contém todos os itens e é para lá que vão ser escritos os itens caso o backend feche e ainda estiverem itens a ser licitados.

Como dito anteriormente, o *Makefile* contém todos os *targets* de execução pedidos.

Os ficheiros "promotor_oficial" e "black_friday" são ficheiros feitos pelos professores que aplicão promoções aos itens.

Por fim, o ficheiro "backend.c" é responsável pela leitura de comandos de administração,verificando e validando cada parâmetro recebido. É o backend que trata de todos os comandos e trata de enviar a mensagem para o frontend.

6. Estruturas de Dados

Clientes

Estrutura que representa os Clientes

```
typedef struct Clientes{
    pid_t pid;
    char nome[TAM_MAX];
    char password[TAM_MAX];
    char comando[TAM_MAX];
    int saldo;
    int hBeat;
    int is_logged_in;
    int tempo_log;
} Clientes;
```

- Pid, id do processo
- nome[TAM_MAX], nome do cliente
- password[TAM_MAX],password do cliente
- comando[TAM_MAX], commando do cliente
- saldo, saldo
- hbeat, HeartBeat
- is_logged_in, se está logado ou não
- tempo_log, tempo do login até ser kickado por inatividade

Promotor

Estrutura que representa os Promotores

```
typedef struct Promotor{
    char message[TAM_MAX];
    char categoria[TAM_MAX];
    int desconto;
    int duracao;
} Promotor;
```

- message[TAM_MAX], messagem do promotor
- categoria[TAM_MAX], categoria do promotor
- desconto, desconto da promoção
- duracao, duração da promoção

Itens

Estrutura que representa os Itens

```
typedef struct Itens
{
    int id;
    char nome[TAM_MAX];
    char categoria[TAM_MAX];
    int preco_base; //valor
    int comprar_ja;
    int tempo;
    char nomeV[TAM_MAX];
    char nomeC[TAM_MAX];
} Itens;
```

- id, id do item
- nome[TAM_MAX], nome do item
- categoria[TAM_MAX], categoria do item
- preco_base, preço base do item
- Comprar_ja, preço da compra instantanea
- Tempo, duração do leilão
- nomeV[TAM_MAX], nome do Vendedor
- nomeC[TAM_MAX] nome do Comprador

ambientVars

Estrutura que representa as variáveis de ambiente

- FPROMOTERS, indica qual o ficheiro do promotor
- FUSERS, Ficheiro dos users
- FITEMS, Ficheiro dos itens
- HEARTBEAT, tempo do HeartBeat

```
typedef struct ambientVars{
    char* FPROMOTERS;
    char* FUSERS;
    char* FITEMS;
    int HEARTBEAT;
}ambientVars;
```

Backend

Estrutura que representa o backend

- Clientes, ponteiro dos clientes
- Itens, ponteiro dos itens
- aVars, ponteiros das variáveis de ambiente
- msg[100], mensagem do backend
- tempo_run, conta o tempo do backend

```
typedef struct Backend{
    Clientes* clientes;
    Itens* itens;
    ambientVars* aVars;
    char msg[100];
    int tempo_run;
    pthread_mutex_t m;
}Backend;
```

dataMSG

Estrutura da mensagem

- pid, PID do processo
- hBeat, HeartBeat
- msg[TAM_LIST], mensagem do frontend

```
typedef struct {
    pid_t pid;
    int hBeat;
    //int clienteSaldo;
    char msg[TAM_LIST];
}dataMSG;
```

7. Verificação, validação e outras questões

7.1 Funcionalidades Realizadas

Considerando o trabalho como um todo, foram implementadas todas as funcionalidades menos tudo o que tenha haver com os promotores

7.2 Observações e Conclusões Finais

A realização deste projeto foi um desafio trabalhoso, mas bastante esclarecedor. O objetivo foi, desde início, uma implementação que ambas entendessem e conseguissem manipular para que o trabalho e esforço individual de cada uma fosse o menos díspar possível. Existem algumas funcionalidades que poderiam ter sido implementadas no projeto de maneira mais eficiente e outras que precisariam de uma melhoria significativa.

Contudo, o trabalho serviu fundamentalmente para a aprendizagem de ambos os elementos do projeto, fornecendo uma ajuda considerável para o exame final da Unidade Curricular.