

**Programação**

Relatório do trabalho prático



Gerenciados de Leilões – Meta 2

Guilherme Camacho – 2021138502

Henrique marques - 2021141476

2022-2023

Índice

Proposta de Trabalho 2

Bibliotecas usadas2

Bibliotecas nativas C 2  
Biblioteca frontend.h3

Biblioteca backend.h3

Biblioteca promotores.h3

Explicação do código frontend.c3

Explicação do código backend.c4

Explicação da falta de existência dos promotores5

**Proposta de trabalho**

Neste trabalho prático foi proposto fazer uma plataforma para gerir leilões em C para a plataforma Unix (Linux). Isto consistes em um servidor (que chamamos por backend) que comunica com vários clientes (que chamamos por frontend) para simular um leilão e também promoções (que chamamos por promotores) que vao alterar os valores dos items que estaos ser leiluados. Base de comunicação entre o frontend e backend é via fifos e thread ondes os fifos são os tubos de comunicações entre um dos frontends e o backend e as thread servem para a sua leitura e escritas das mensagem que ficam a vaguear por esse tubos. Os promotores (que não estão neste trabalho, pelo motivo na parte final do relatório) não podia ser usado pelos pipes.

**Bibliotecas usadas**

**Bibliotecas nativas do C**

Foi utilizados as bibliotecas *stdio.h*, *stdlib.h*, *string.h* e *ctype.h.*

Stdio.h – biblioteca de input e output na consola

Stdlib.h – biblioteca para a função malloc() para alocar informação na memoria

String.h – biblioteca para manipuçar as strings com facilidade

Ctype.h – biblioteca usada para tirar o caps das strings

Unistd.h – biblioteca que fornece acesso à API do Linux

Fcntl.h – biblioteca para o controlo de ficheiros

Errno.h – biblioteca para os erros

Signal.h – biblioteca para os sinais

Pthread.h – biblioteca para as threads

Sys/types.h – biblioteca para os tipos dfe dados

Sys/Stat.h – biblioteca para ter os status do ficheiro

**Biblioteca frontend.h**

Foi criada com o propósito de armazenar a estrutura do clientes.

Frontend – estrutura que armazena o PID do cliente, o seu saldo, o seu username, a sua password, o nome do seu pipe e por fim o comando que manda para o backend. Esta estrutura também se chama *cliente*.

**Biblioteca backend.h**

Foi criada com o propósito de armazenar a estrutura do servidor.

Backend - estrutura que armazena a variável de tempo, o heartBeat, o numero máximo de clientes, o numero máximo de promotores, o numero de clientes ligados/ativos, o numero de promotores ligados/ativos, uma lista de clientes e uma lista de promotores. Esta estrutura também se chama *server*.

**Biblioteca prmotor.h**

Foi criada com o propósito de armazenar a estrutura do promotor.

Promotor - estrutura que armazena a categoria do promotor, o seu desconto e a sua duração. Esta estrura também se chama *discount* e um ponteiro *discount*\_prt.

**Explicação do código do frontend.c**

Começa por definir os nomes dos ficheiros fifos, as mensagems que pode receber na validação de entrada para o backend e a mensagem para paramentro de comandos invalidas. Depois são decladas variáveis globais.

A função *fecharFrontend()* tem o propósito de, quando o frontend recebe um sinal para fechar, ele fecha os tuner par enviar e receber para o backend e apaga o ficheiro fifo.

A função *\*readMensagem(void \*vargp)* tem o propósito para a rotina da thread dedica a ler as mensagens vinda do backend. Ela começa por verificar se o tunel está bem construído e depois entra em loop para receber as mensagens do backend até o programa mandar para.

A função *\*writeMensagem(void \*vargp)*tem o propósito para a rotina da thread dedica a enviar os comandos do frontend para o backend. Ela começa por verificar se o tunel está bem construído e depois entra em loop para mandar os comandos que fica à espera do cliente.

A função *backendAberto()* tem o propósito de vericar se o backend esta aberto. Se não tiver, o frontend avisa que o backend não esta aberto e para.

No main, ele recebe as credenciais pelos os argumentos na execução do programa, verifica se o backend esta aberto, verifica se foram inseridas as credencias e começa a configurar a estrutura cliente para mandar as credencias e seus comandos, as threads com as suas respetivas rotinas.

**Explicação do código do backend.c**

Começa por definir os nomes dos ficheiros fifos, as mensagems que pode receber na validação de entrada para o enviar para o frontend a mensagem para paramentro de comandos invalidas. Depois são decladas variáveis globais, define a estura items para guardar os itens.

A função *backendAberto()* verifica se já exite um backend aberto basedo na existe de um fifo.

A função *loadItemFile(char \*pathname, item \*\*list)* no carregamento do ficheiro de itens numa lista de itens. E *saveItemFile(char \*pathname, item \*\*list, int listLen)* em salvar os items no ficheiro.

A funçao *\*tunnelUser()* serve para rotina do thread fica a “falar” com os clientes, onde verifica as credencias e os seus comandos.

A função *\*timerItem()* serve como rotina de temporizador onde verifica se os clientes estão ativos (se o cliente não saiu sem mandar o comando de sair), o tempo do item que está a ser leiluado e a contar o tempo que o servidor esta aberto.

A função *\*consoleAdmin()* serve como rotina para ver as mensagems do administrador.

No main, vais buscar as variáveis ambiente, configura os servidor, verifica se já existe um backend abarto, carrga os ficheiros FUSERS e FITEMS, cria os fifos de comunicação e as threads com as suas respetivas rotinas.

**Explicação da falta de existência dos promotores**

Por motivos de saúde, um membro do grupo não pode ajudar na criação dos promotores e, pela a falta de conhecimento, não foi alcaçando o objetivo de fazer os promoteres serem criados a tempo.

Pedimos desculpa por o nosso trabalho não às mil maravilhas, mas quisemos entregar para demostrar as nossas capacidades, mesmo com a falta de disponibilidade para trabalhar.