Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra
Instituto Politécnico de Coimbra

Licenciatura em Engenharia Informática

Unidade Curricular de Sistemas Operativos

Ano Letivo de 2022/2023

Programação em C para UNIX

Sistema de Leilões

Meta 1

Nº 2019135835 – Henrique Barradas

Nº2019131769 – João Carvalho

Índice

Introdução

No âmbito da cadeira de Sistemas Operativos, ao longo do semestre vai ser implementado um sistema para gerir a venda de itens em vários leilões, que encaminha e medeia a interação entre os clientes e o administrador. Este trabalho destina-se a correr em ambiente Unix linha de comandos e irá mediar a interação entre doente, médico e balcão de atendimento.

Na primeira meta vamos falar um pouco sobre as estruturas de dados, das variáveis de ambiente e do funcionamento dos pipes.

Estruturas de Dados

Clientes

- **nome:** string onde é guardada o nome do cliente;
- **password:** string onde é guardada a password do cliente;
- **saldo:** inteiro que serve para guardar o saldo do cliente.

```
typedef struct Clientes{
    char nome[TAM];
    char password[TAM];
    int saldo;
} Clientes, *ptrClientes;
```

Figura 1

Promotor

- **Foi usada apenas para testes**

```
typedef struct Promotor{
    char message[TAM];
    char categoria[TAM];
    int desconto;
    int duracao;
} Promotor, *ptrPromotor;
```

Figura 2

Itens

- **Id:** guarda o id do item
- **Nome:** guarda o nome do item
- **Categoria:** guarda a categoria do item
- **Preco_base:** guarda o preço inicial
- **Comprar_ja:** guarda o preco de compra instantânea
- **Tempo:** tempo do leilão:
- **NomeV:** Nome do Vendedor
- **NomeC**: Nome do comprador

```
typedef struct Itens
{
    int id;
    char nome[TAM];
    char categoria[TAM];
    int preco_base; //valor a ser incrementado
    int comprar_ja;
    int tempo;
    char nomeV[TAM];
    char nomeC[TAM];
} Itens, *ptrItens;
```

Figura 3

Variáveis ambiente

- **FPROMOTERS:** arrray que guarda o nome do ficheiro
- **FUSERS:** array que guarda o nome do ficheiro
- **FITENS:** array que guarda o nome do ficheiro

```
typedef struct ambientVars{
    char* FPROMOTERS;
    char* FUSERS;
    char* FITEMS;
}ambientVars, *ptrAmbientVars;
```

Figura 4

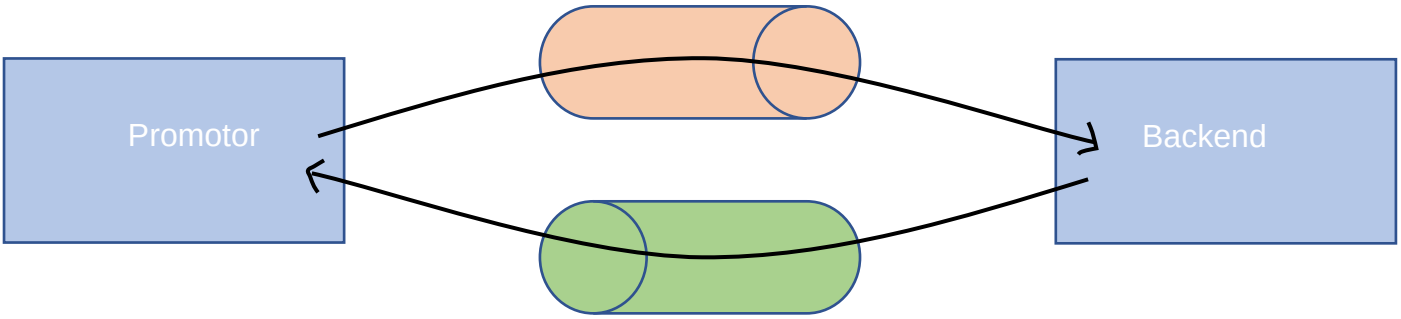
Estrutura dos pipes

- Estrutura que guarda o pipe responsavel pela comunicacao entre o promotor e o back end

```
typedef struct HandlerPromotor{
    int fd[2];
} HandlerPromotor, *ptrHandlerPromotor;
```

Figura 5

Unnamed Pipes



Na parte da comunicação entre o backend e o promotor, utilizamos os “unnamed pipes”. Neste pedaço de código mostramos como é feito a escrita e a leitura pelo processor “pai” para que seja possível enviar a mensagem ao Promotor. Como mostramos na figura em cima, é utilizado um pipe para a escrita e um pipe para a leitura.

Na figura 6 é possível verificar o método de implementação dos pipes.

```
ptrHandlerPromotor openPromoter(ptrHandlerPromotor pP, ptrAmbientVars aVars){

    char msgPromotor[TAM];
    char path[100];
    char ff[TAM] = "../promotor_files/";
    char fff[TAM] = "./";

    strcpy(path, strcat(ff, aVars->FPROMOTERS));
    strcat(fff, aVars->FPROMOTERS);

    pipe(pP->fd);

    int id = fork();

    if(id < 0){
        printf("[ERRO] Promotor nao foi criado com sucesso\n");
        return NULL;
    }else if(id == 0){
        close(1); //fecha o stdout no file descriptor
        dup(pP->fd[1]); //duplica o stdout
        close(pP->fd[0]); //fecha o antigo
        close(pP->fd[1]); // fecha a outra ponta do pipe

        execl(path, fff, NULL);
    }else if(id > 0){
        read(pP->fd[0], msgPromotor, sizeof(msgPromotor)); //lê o que recebe do printf do promotor atraves do pipe
        close(pP->fd[1]); //fecha a ponta do pipe onde foi escrito
        printf("%s", msgPromotor); //printa a mensagem do promotor

        /*union sigval xpto;
        sigqueue(id, SIGUSR1, xpto);*/

        //working
        kill(id, SIGKILL);
        wait(&id);

        return 0;
    }

    return pP;
}
```

Figura 6