Relatório SO Meta 2

9 de janeiro de 2023

Trabalho realizado por:
Carlos Pinto | a2021155627
Alexandre Ferreira |a2021138219

Introdução

Este relatório irá descrever a estratégia adotada na resolução do trabalho proposto na cadeira de Sistemas Operativos que tem como objetivo a criação de um programa de leilões com várias funcionalidades.

Desenvolvimento

Existem dois tipos de utilizadores que podem aceder a este programa, o cliente e o administrador, o cliente acede a partir do programa frontend e o administrador a partir do backend, para o cliente aceder ao programa é necessário o backend estar em execução, caso contrário era impossível comunicar visto que podem estar conectados múltiplos frontends e apenas um backend, para certificar que o programa backend está em execução é usada a função access() que retorna um valor consoante a disponibilidade do named pipe, este named pipe é apenas acessível caso o backend esteja em execução, o backend pode ainda receber input do administrador com a ajuda de um select() que alterna entre o named pipe e o input do administrador, quando um utilizador se conecta é usado esse mesmo named pipe para “avisar” o backend que um usuário se conectou, o backend devolve um id único para o frontend que é essencial na comunicação entre o backend e o frontend visto que vários frontends se podem conectar.

```
struct mensagem{
    char texto[TAMMSG];
    int valor;
};
```

Figura 1Figura Estrutura usada na comunicação

```
struct utilizador{
    char username[TAM],password[TAM];
    int saldo,id;
};


struct item{
    int id;
    char nome[TAM];
    char categoria[TAM];
    int valor;
    int valorCompreJa;
    int duracao;
    char usernameVende[TAM];
    char usernameCompra[TAM];
};
struct promotor{
    char categoria[20];
    int desconto,duracao;

};
```

Figura 2Outras estruturas essencias

```
char pipeid[10];
sprintf( s: pipeid,  format: "%d", *(int *) arg);
char rdpipe[TAM];
char wrpipe[TAM];
strcat( dest: pipeid,  src: "rd");
strcpy( dest: rdpipe,  src: pipeid);
strcat( dest: pipeid,  src: "wr");
strcpy( dest: wrpipe,  src: pipeid);

int wr = open( file: rdpipe,  oflag: O_RDWR);
int rd = open( file: wrpipe,  oflag: O_RDWR);
```

Figura 3Excerto de código

A partir do id devolvido ao utilizador são criados dois named pipes únicos por cada utilizador (um para escrever e outro para ler, para facilitar a comunicação), é neste FIFO que o frontend envia as credenciais ao backend, que faz a validação (com o auxílio da biblioteca fornecida) e retorna uma mensagem com o resultado,

Caso bem sucedida o utilizador poderá então usufruir das funcionalidades do programa, como ver o tempo de execução do programa.

O tempo de execução começa a contar assim que o programa backend é inicializado, com o auxilio de mais uma thread e a função wait().

Existe também uma thread que verifica o prazo de todos os items a serem vendidos, e assim que o prazo expira, o item, caso tenha um licitador fica na posse do mesmo, caso contrário deixa apenas de ser possível comprar.

A execução dos promotores é feita a pedido do administrador e é criada uma thread para esse fim.

Todos os itens (e utilizadores) são mantidos em vetores para facilitar a sua manutenção, estes vetores são atualizados com o auxilio de uma função (LerItems()) que obtém esses valores através de um ficheiro .txt que contém todas as informações necessárias, estes vetores são uma mais valia quando, por exemplo, se precisa de mostrar todos os itens ao utilizador .

Quando um administrador usa a função kick() o usuário indicado (caso conectado) é expulso com uma mensagem especial.

Assim que o administrador invocar a função close todos os utilizadores conectados perderão acesso ao programa, é feito o unlink dos pipes e o programa encerra, guardando em ficheiros os itens e utilizadores.

Conclusão

Com este trabalho, melhoramos as nossas aptidões e conhecimento em partes essenciais no que toca à programação tais como o uso de threads, select e sinais, manipulação de ficheiro, e raciocínio lógico.