Instituto Superior de Engenharia de Coimbra Licenciatura em Engenharia Informática

Sistemas Operativos 2

Trabalho Prático Meta 2-

Relatório

Bruno André Rodrigues Cancela – 2020131288 LEI

Coimbra, 19 de maio de 2024

isec Engenharia

Índice

1. Introdução

Este relatório descreve a implementação do sistema de bolsa de valores online desenvolvido no âmbito da disciplina de Sistemas Operativos 2. O trabalho prático consistiu na criação de vários programas que, em conjunto, simulam uma bolsa de valores simplificada. Este sistema permite operações de compra e venda de ações entre utilizadores através de interfaces de consola.

2. Arquitetura do Sistema

O sistema é composto pelos seguintes programas:

- Bolsa: O servidor central que gere todas as operações e dados do sistema.
- Cliente: A interface do utilizador que permite aos usuários interagir com o sistema bolsa.
- Board: Um programa que apresenta uma lista das empresas cujas ações são mais valiosas e os dados da última transação.

3. Mecanismos de Comunicação e Sincronização

Bolsa - Cliente

A comunicação entre o programa Bolsa e o programa Cliente é feita através de named pipes. Cada cliente envia comandos para o Bolsa e recebe respostas através destes pipes.
Para melhorar a eficiência e permitir que o programa Bolsa continue a responder a outras comandos enquanto espera pela conclusão das operações de leitura/escrita, utilizamos Overlapped I/O, uma técnica que permite que operações de entrada/saída (I/O) sejam executadas de maneira assíncrona, em vez de bloquear o processo.

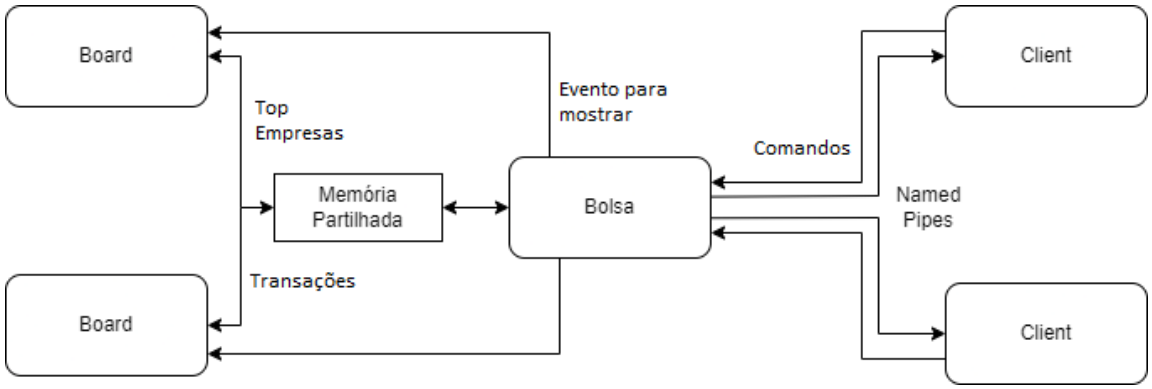
Bolsa - Board

A comunicação entre o programa Bolsa e o programa Board é realizada através de memória partilhada. A Bolsa atualiza a memória partilhada com os dados necessários e utiliza eventos para notificar o Board sobre as novas informações a serem mostradas.

Para garantir que a Board mostra informações atualizadas em tempo real, como as empresas mais valiosas e as últimas transações, utilizamos eventos. Estes eventos notificam o Board sempre que há novos dados disponíveis, evitando a necessidade de leitura periódica.

Para evitar a corrupção de dados devido ao acesso concorrente, utilizamos mutexes. Os mutexes asseguram que apenas um processo de cada vez pode aceder e modificar a memória partilhada, garantindo a integridade dos dados.

Diagrama de Comunicação



O diagrama ilustra a comunicação entre os diferentes componentes do sistema de bolsa de valores, utilizando named pipes e memória partilhada.

Componentes Principais

- Bolsa, Client, Board

Comunicação Bolsa - Cliente

- Named Pipes:
    - A comunicação entre a Bolsa e os Clients é feita através de named pipes, representada por setas bidirecionais.
    - Os Clients enviam comandos para a Bolsa e recebem respostas através destes canais.

Comunicação Bolsa – Board

- Memória Partilhada:
    - A comunicação entre a Bolsa e os Boards é realizada através de memória partilhada.
    - A Bolsa atualiza a memória partilhada com informações sobre as empresas mais valiosas (Top Empresas) e as transações realizadas.
    - Eventos: São utilizados para notificar os Boards sobre novos dados a serem exibidos, garantindo que as informações sejam atualizadas em tempo real.

Fluxo de Dados

- Top Empresas: Dados sobre as empresas mais valiosas são escritos na memória partilhada pela Bolsa e lidos pelos Boards.
- Transações: Informações sobre as transações são escritas na memória partilhada pela Bolsa e lidas pelos Boards.
- Comandos: Os Clients enviam comandos para a Bolsa, que processa esses comandos e envia as respostas de volta através dos named pipes.

Sincronização

- Mutexes: São utilizados para garantir a integridade dos dados na memória partilhada, assegurando que apenas um processo por vez pode aceder e modificar os dados.

4. Estruturas de Dados

Estrutura de Dados do Programa Bolsa

```c
typedef struct _Empresa {
    TCHAR nome[BUFFER_SIZE];
    unsigned int acoes;
    unsigned int acoesDisponives;
    float preco;
} Empresa;

typedef struct _Mensagens {
    Empresa empresas[MAX_EMPRESAS];
    TCHAR mensagem[BUFFER_SIZE];
    unsigned int tipo;
    unsigned int nEmpresas;
} Mensagens;

typedef struct _Carteira {
    Empresa* empresa;
    unsigned int nAcoesCompradas;
    unsigned int nAcoesParaVenda;
} Carteira;

typedef struct _Cliente {
    TCHAR nome[BUFFER_SIZE];
    TCHAR password[BUFFER_SIZE];
    float saldo;
    unsigned int estado;
    unsigned int nCarteiraEmpresas;
    Carteira carteira[5];
} Cliente;

typedef struct _TopEmpresas {
    Empresa empresas[MAX_TOP_EMPRESAS];
    int nEmpresas;
} TopEmpresas;

typedef struct _Transacao {
    TCHAR nome[BUFFER_SIZE];
    int nAcoes;
    float valorTotal;
} Transacao;

typedef struct SharedMessage {
    TopEmpresas topEmpresas;
    Transacao transacao;
} SharedMessage;
```

```c
typedef struct _SharedMemory {
    HANDLE hMapFile; //createfilemaping
    SharedMessage* sharedMessage;
    //utlima transação
    int threadMustContinue; //trinco
    HANDLE CompEvent; //evento que sin
    HANDLE TransacEvent; //evento que
    HANDLE CloseBoardEvent;
    HANDLE hRWMutex;
} SharedMemory;

typedef struct _ControlData {
    unsigned int shutdown;  // flag "c
    HANDLE alredyRunning;
    Empresa empresas[MAX_EMPRESAS];
    Cliente clientes[MAX_CLIENTES];
    HANDLE pipeClientes[MAX_CLIENTES];
    unsigned int nEmpresas;
    unsigned int maxClientes;
    unsigned int nClientes;
    unsigned int pause;
    HANDLE hSemaphore;
    HANDLE CloseEvent;
    SharedMemory sharedMemory;
    HANDLE hMutexBolsa;
} ControlData;

typedef struct _ThreadClient {
    HANDLE* hPipe;
    ControlData* controlData;
} ThreadClient;
```

- Empresa: Armazena informações sobre uma empresa, incluindo nome, total de ações, ações disponíveis e preço das ações.
- Mensagens: Utilizadas para enviar dados da Board para o Cliente, contendo um array de empresas, uma mensagem, o tipo de mensagem e o número de empresas.
- Carteira: Armazena as ações que um cliente possui de uma empresa específica, incluindo o número de ações compradas e disponíveis para venda.
- Cliente: Armazena informações sobre um cliente, incluindo nome, senha, saldo, estado, número de empresas na carteira e um array de carteiras.

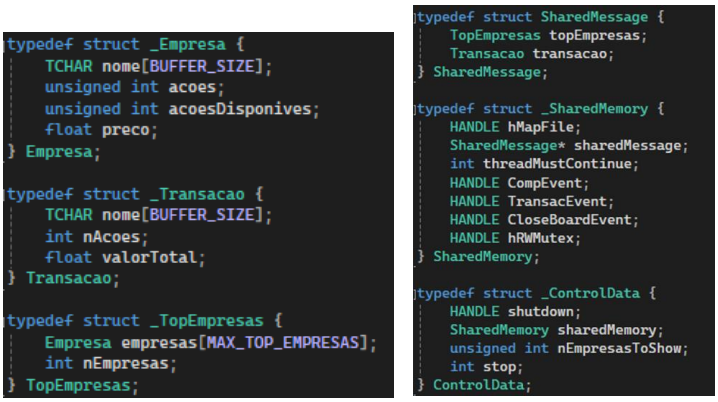
- TopEmpresas: Armazena as empresas mais valiosas, com um array de empresas e o número total de empresas.
- Transacao: Armazena informações sobre uma transação, incluindo o nome da empresa, número de ações e valor total da transação.
- SharedMessage: Utilizada para partilhar informações entre processos, contendo as empresas mais valiosas e a última transação.
- SharedMemory: Gerência a memória partilhada entre processos, com handles para sincronização e acesso à memória.
- ControlData: Centraliza informações de controle e gerenciamento, incluindo empresas, clientes, pipes, eventos e memória partilhada.
- ThreadClient: Passa informações para threads que lidam com clientes, incluindo o pipe do cliente e dados de controle.

Estrutura de Dados do Programa Cliente

```
typedef struct _Empresa {
    TCHAR nome[BUFFER_SIZE];
    unsigned int acoes;
    unsigned int acoesDisponives;
    float preco;
} Empresa;

typedef struct _Mensagens {
    Empresa empresas[MAX_EMPRESAS];
    TCHAR mensagem[BUFFER_SIZE];
    unsigned int tipo;
    unsigned int nEmpresas;
} Mensagens;


#define TOPSIZE sizeof(TopEmpresas)

typedef struct _ControlData {
    unsigned int shutdown;
    HANDLE alredyRunning;
    HANDLE hPipe;
    HANDLE hSemaphore;
    int loggedIn;
    int bolsaClosed;
} ControlData;
```

- ControlData: Centraliza informações de controle e gerenciamento, incluindo flags de controle, handles para comunicação e semáforo, e estado de login e fechamento da bolsa.

Estrutura de Dados do Programa Board



```
typedef struct _Empresa {
    TCHAR nome[BUFFER_SIZE];
    unsigned int acoes;
    unsigned int acoesDisponives;
    float preco;
} Empresa;

typedef struct _Transacao {
    TCHAR nome[BUFFER_SIZE];
    int nAcoes;
    float valorTotal;
} Transacao;

typedef struct _TopEmpresas {
    Empresa empresas[MAX_TOP_EMPRESAS];
    int nEmpresas;
} TopEmpresas;
```

```
typedef struct SharedMessage {
    TopEmpresas topEmpresas;
    Transacao transacao;
} SharedMessage;

typedef struct _SharedMemory {
    HANDLE hMapFile;
    SharedMessage* sharedMessage;
    int threadMustContinue;
    HANDLE CompEvent;
    HANDLE TransacEvent;
    HANDLE CloseBoardEvent;
    HANDLE hRWMutex;
} SharedMemory;

typedef struct _ControlData {
    HANDLE shutdown;
    SharedMemory sharedMemory;
    unsigned int nEmpresasToShow;
    int stop;
} ControlData;
```

- ControlData: Centraliza informações de controle e gerenciamento, incluindo o handle de shutdown, a estrutura de memória partilhada, número de empresas a mostrar e flag para parar operações.

5. Implementação de Funcionalidades

Programa Bolsa

```
void initReg(ControlData* cd);
BOOL initMemAndSync(ControlData* cdata);
BOOL loadUsers(ControlData* cdata);
void companiesAndTransactions(ControlData* cdata);
void checkTop10(ControlData* cdata);
void sortCompanies(ControlData* cdata);
void HandleAddCompany(ControlData* cd, const TCHAR* params);
BOOL loadCompanies(ControlData* cdata);
void WriteToClient(HANDLE hPipe, const Mensagens* message);
void HandleListCompanies(ControlData* cd);
void HandleSetStockPrice(ControlData* cd, const TCHAR* params);
void HandleListUsers(ControlData* cd);
void HandlePauseOperations(ControlData* cd, const TCHAR* params);
void HandleClosePlatform(ControlData* cd);
void Broadcast(ControlData* cd, Mensagens mensagem);
int ProcessCommand(ControlData* cd, TCHAR* command);
void ProcessClientCommand(LPTSTR buffer, ControlData* cd, Cliente* currentClient, HANDLE hPipe);
DWORD WINAPI ReadFromClient(LPVOID lpParam);
DWORD WINAPI CreatePipeAndHandleClients(LPVOID lpParam);
DWORD WINAPI ThreadPause(LPVOID lpParam);
```

- Adicionar Empresa: Permite adicionar uma nova empresa com nome, número de ações e preço por ação.
- Listar Empresas: Lista todas as empresas registadas, mostrando o nome, ações disponíveis e preço atual.
- Redefinir Custo de Ações: Redefine o preço das ações de uma empresa existente.
- Listar Utilizadores: Lista todos os utilizadores, mostrando nome, saldo e estado (online/offline).

- Pausar Operações: Pausa todas as operações de compra e venda por um tempo especificado.
- Encerrar Plataforma: Fecha a plataforma e notifica todos os clientes e boards conectados.
- Carregar Empresas de um Ficheiro: Carrega informações de empresas a partir de um ficheiro e adiciona ao sistema.
- Carregar Utilizadores de um Ficheiro: Carrega informações de utilizadores a partir de um ficheiro e adiciona ao sistema.
- Autenticar Utilizador: Verifica nome de utilizador e senha, autenticando o utilizador se as credenciais forem válidas.
- Processar Comandos de Clientes: Recebe e processa comandos enviados pelos clientes através dos named pipes.

Programa Cliente

```
HANDLE ConnectToServer();
DWORD WINAPI ReadFromServer(LPVOID lpParam);
void WriteToServer(HANDLE hPipe, const TCHAR* message);
int HandleLogin(ControlData* cd, TCHAR* command);
int HandleBuy(ControlData* cd, TCHAR* command);
int HandleSell(ControlData* cd, TCHAR* command);
int ProcessCommand(ControlData* cd, TCHAR* command);
```

- Autenticar Utilizador: Permite ao utilizador fazer login com nome de utilizador e senha.
- Listar Empresas: Lista todas as empresas disponíveis na bolsa.
- Comprar Ações: Permite ao utilizador comprar ações de uma empresa.
- Vender Ações: Permite ao utilizador vender ações da sua carteira.
- Consultar Saldo: Mostra o saldo atual do utilizador.
- Sair da Aplicação: Encerra a sessão do utilizador e desconecta do servidor.

Programa Board

```
BOOL initMemAndSync(ControlData* cdata);
DWORD WINAPI topCompanies(LPVOID p);
DWORD WINAPI showTransanction(LPVOID p);
```

- Apresentar Empresas Mais Valiosas: Mostra uma lista das empresas mais valiosas em tempo real.
- Apresentar Última Transação: Mostra detalhes da última transação realizada na bolsa.

- Sincronização com Memória Partilhada: Sincroniza dados de empresas e transações usando memória partilhada e eventos.

6. Tabela de Requisitos Implementados

Bolsa

ID	Descrição da Funcionalidade / Requisito	Estado	Detalhes
1	Adicionar uma empresa	Implementado	Permite adicionar uma nova empresa com nome, número de ações e preço por ação. Utiliza a função **HandleAddCompany**.
2	Listar todas as empresas	Implementado	Lista todas as empresas registadas, mostrando o nome, ações disponíveis e preço atual. Utiliza a função **HandleListCompanies**.
3	Redefinir custo das ações	Implementado	Redefine o preço das ações de uma empresa existente. Utiliza a função **HandleSetStockPrice**.
4	Listar utilizadores	Implementado	Lista todos os utilizadores, mostrando o nome, saldo e estado (online/offline). Utiliza a função **HandleListUsers**.
5	Pausar operações	Implementado	Pausa todas as operações de compra e venda por um tempo especificado. Utiliza a função **HandlePauseOperations**.
6	Encerrar a plataforma	Implementado	Fecha a plataforma e notifica todos os clientes e boards conectados. Utiliza a função **HandleClosePlatform**.
7	Carregar empresas de um ficheiro	Implementado	Carrega informações de empresas a partir de um ficheiro e adiciona ao sistema. Utiliza a função **loadCompanies**.
8	Carregar utilizadores de um ficheiro	Implementado	Carrega informações dos utilizador a partir de um ficheiro e adiciona ao sistema. Utiliza a função **loadCompanies**.
9	Autenticar utilizador	Implementado	Verifica nome de utilizador e senha, autenticando o utilizador se as credenciais forem válidas. Utiliza a função **ProcessClientCommand** para autenticação.
10	Processar comandos de clientes	Implementado	Recebe e processa comandos enviados pelos clientes através dos named pipes. Utiliza a função **ProcessClientCommand**.
11	Obter valor máximo de utilizadores ligados	Implementado	Obtém o número máximo de utilizadores logados a partir de uma chave do registo do Windows. Utiliza a função **InitReg.**
12	Sincronização com memória partilhada	Implementado	Sincroniza dados de empresas e transações usando memória partilhada e eventos. Utiliza a função **initMemAndSync**.

Cliente

ID	Descrição da Funcionalidade / Requisito	Estado	Detalhes
1	Autenticar utilizador	Implementado	Permite ao utilizador fazer login com nome de utilizador e senha. Utiliza a função **HandleLogin**.
2	Listar empresas	Implementado	Lista todas as empresas disponíveis na bolsa. Utiliza a função **ProcessCommand** com o comando **listc**.
3	Comprar ações	Implementado	Permite ao utilizador comprar ações de uma empresa. Utiliza a função **HandleBuy**.
4	Vender ações	Implementado	Permite ao utilizador vender ações de sua carteira. Utiliza a função **HandleSell**.
5	Consultar saldo	Implementado	Exibe o saldo atual do utilizador. Utiliza a função **ProcessCommand** com o comando **balance**.
6	Sair da aplicação	Implementado	Encerra a sessão do utilizador e desconecta do servidor. Utiliza a função **ProcessCommand** com o comando **exit**.

Board

ID	Descrição da Funcionalidade / Requisito	Estado	Detalhes
1	Apresentar empresas mais valiosas	Implementado	Mostra uma lista das empresas mais valiosas em tempo real. Utiliza a função **topCompanies**.
2	Apresentar última transação	Implementado	Mostra detalhes da última transação realizada na bolsa. Utiliza a função **showTransanction**.
3	Sincronização com memória partilhada	Implementado	Sincroniza dados de empresas e transações usando memória partilhada e eventos. Utiliza a função **initMemAndSync**.

7.Conclusão

A implementação do sistema de bolsa de valores online permitiu explorar diversos conceitos de sistemas operativos, como comunicação entre processos, sincronização e gerenciamento de memória partilhada. Consegui implementar todas as funcionalidades pedidas no trabalho prático de Sistemas Operativos 2. O sistema de bolsa de valores online foi desenvolvido com sucesso, incluindo a comunicação entre os programas Bolsa, Cliente e Board através de named pipes e memória partilhada.

As funcionalidades exigidas foram cumpridas. No entanto, não consgui implementar a funcionalidade bônus (BoardGUI), que teria permitido a visualização gráfica das informações. Apesar disso, o sistema atende plenamente aos requisitos principais e proporciona uma experiência funcional para a simulação de uma bolsa de valores online.