INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

INSTITUTO POLITÉCNICO DE COIMBRA

Trabalho Prático – 2ªMeta
Bolsa de valores online

João Duarte Nº 2020122715
André Dias Nº2021140917

COIMBRA

18 de maio de 2024

Índice

Introdução

O trabalho prático consiste na implementação de vários programas que, no seu conjunto, constituem uma bolsa de valores online. A bolsa de valores tem um comportamento, apesar de simplificado, semelhante ao conceito comum de bolsa de valores. Na bolsa são transacionadas ações de empresas. As empresas, as ações disponíveis para compra/venda e o preço de cada ação são controlados de forma centralizada.

No âmbito do trabalho, todos os intervenientes serão **processos** a correr na **mesma máquina** e apenas é necessária a comunicação entre processos.

O trabalho prático foi realizado em linguagem C, usando a API Win32 do Windows.

Arquitetura do Sistema

Aplicação bolsa:
Responsável pela gestão centralizada dos dados e das operações na bolsa de valores. Implementa as funcionalidades descritas no enunciado, como adicionar empresas, listar empresas, atualizar preços de ações, listar utilizadores, pausar operações e encerrar a plataforma.
Comunica com os clientes utilizando named pipes e com a aplicação board utilizado memória partilhada.

Aplicação cliente:
Interface de utilizador para interação com o sistema da bolsa.
Permite que os utilizadores se autentiquem, consultem empresas, comprem e vendam ações e verifiquem os seus saldos.
Comunica com a bolsa de valores utilizando names pipes.

Aplicação board:
Apresenta uma lista das empresas mais valiosas, por ordem decrescente de valor e informações sobre a última transação.
Comunica com a bolsa de valores utilizando memória partilhada.

Utils:
Possui todas as estruturas, constantes e “includes” necessários para guardar e definir a informação necessária para o funcionamento da aplicação.

Estruturas de dados

Ações

```
typedef struct {
    TCHAR empresa[20];
    DWORD qtAcoes;
}Acoes;
```

Representa a informação sobre as ações de uma empresa. Contém o nome da empresa (empresa) e a quantidade de ações (qtAcoes) dessa empresa que um utilizador possui.

User

```
typedef struct {
    TCHAR username[MAX_TAM];
    TCHAR password[20];
    float saldo;
    BOOL login;
    HANDLE hPipe;
    DWORD dwThreadID;
    int nAcoes;
    Acoes carteira[5]; //........ associar as acoes de compra
} User;
```

Contém informações sobre um utilizador do sistema. Inclui o nome de utilizador (username), a palavra-passe (password), o saldo (saldo) da conta do utilizador, o estado de login (login), o identificador do pipe (hPipe), o identificador da thread (dwThreadID), o número de ações (nAcoes) que o utilizador possui e uma matriz (carteira) que associa as ações de compra do utilizador.

Empresa

```
typedef struct {
    TCHAR nome[100];
    float precoAcao;
    DWORD acoesDisponiveis;
}Empresa;
```

Representa uma empresa. Inclui o nome da empresa (nome), o preço por ação (precoAcao) e a quantidade de ações disponíveis (acoesDisponiveis) dessa empresa.

Transações

```
typedef struct {
    TCHAR tipo;
    Empresa empresa;
    User* user;
    DWORD numAcoes;
    float precoAcoes;
} Transacoes;
```

Guarda informações sobre uma transação. Inclui o tipo de transação (tipo), a empresa envolvida (empresa), o utilizador associado (user), o número de ações (numAcoes) envolvidas na transação e o preço das ações (precoAcoes).

SharedData

```
typedef struct {
    Empresa empresas[MAX_EMPRESAS];
    int numEmpresas; //contador de empresas
    Transacoes lastTransacao;
    User users[999];
    HANDLE hPipe;

    BOOL pausedBolsa;
    int seconds;
}SharedData;
```

Mantém os dados partilhados entre diferentes partes do sistema. Inclui um array de empresas (empresas), o número de empresas (numEmpresas), a última transação realizada (lastTransacao), um array de utilizadores (users), um identificador de pipe (hPipe), uma flag para pausar a bolsa (pausedBolsa), e um contador de segundos (seconds).

SharedMemory

```
typedef struct {
    HANDLE hMapFile;
    HANDLE hEventUpdateBoard;
    HANDLE hMutexUpdateBoard;
    HANDLE hEventRunning;

    SharedData* sharedData;

}SharedMemory;
```

Contém identificadores e ponteiros para os recursos partilhados. Inclui identificadores para um ficheiro mapeado (hMapFile), um evento de atualização da board (hEventUpdateBoard), um mutex para atualizar a board (hMutexUpdateBoard), um evento para indicar a pausa da bolsa(hEventRunning) e um ponteiro para dados partilhados (sharedData).

Mecanismos de comunicação

Memória Partilhada

A memória partilhada é utilizada para a comunicação entre o programa bolsa e o programa board permitindo que ambas as aplicações acedam e atualizem os dados como por exemplo os valores das ações e a informação sobre as últimas transações.

A memória partilhada é criada na função "`initSharedMemory_Sync`" em ambos os programas bolsa e board com o nome de "hMapsharedmemory".

Named Pipes

A comunicação entre o cliente e a bolsa vai ser realizada através de named pipes(OVERLAPPED I/O), permitindo a troca de informações como o envio de comandos por parte do cliente para a aplicação "bolsa" e a receção de respostas. A bolsa vai criar um named pipe e receber as entradas e conexões. Este método garante que os dados escritos pelo cliente sejam lidos pela bolsa na mesma ordem, mantendo a integridade das informações.

Mecanismos de sincronização

Mutexes

Para garantir que **apenas existe uma instância** da aplicação bolsa a executar foi utilizado um **mutex** com o nome "mutexBolsa" sendo este também usado para verificar se já existe uma instância da bolsa a executar quando for executada a aplicação board. Caso não exista, esta é terminada. Caso se tentem executar várias bolsas, **apenas a primeira terá sucesso** e as outras serão terminadas.

O **mutex** "hMutex" é usado para proteger **secções críticas** de código onde há acesso e modificação de **dados partilhados** (sharedMemory->sharedData). Ele garante que apenas uma thread de cada vez possa executar essas secções, prevenindo condições de corrida e garantindo a consistência dos dados:

Login e Logout do Utilizador: Protege a verificação e atualização do estado de login dos utilizadores.
Compra de Ações: Assegura que a verificação das condições de compra e a atualização dos dados da empresa e do utilizador sejam completas na totalidade ou não sejam feitas de todo.
Venda de Ações: Garante que a verificação das condições de venda e a atualização dos dados da empresa e do utilizador sejam feitas de forma segura.
Verificação do Saldo: Protege a leitura e possível modificação do saldo do utilizador.

Semáforos

Eventos

Cada vez que é efetuada uma ação que desencadeie uma atualização na board, é utilizada uma função que contém um **evento** por parte da bolsa "hEventUpdateBoard" que vai informar a board que é necessário atualizar. Para isto a board possui uma **thread** que ao receber esse evento vai atualizar a informação apresentada.

O evento "ReadReady" serve para sincronizar a conclusão de uma operação de leitura assíncrona no pipe.
Este mecanismo permite que a thread realize outras tarefas enquanto espera pela conclusão da operação de leitura, melhorando a eficiência e a responsividade do programa.

O evento "WriteReady" serve para sincronizar a conclusão de uma operação de escrita assíncrona no pipe.
Este mecanismo permite que a thread realize outras tarefas enquanto espera pela conclusão da operação de escrita, melhorando a eficiência e a responsividade do programa.

O evento "hEventRunning" serve para sincronizar e coordenar a execução da função de pausa.
Este mecanismo garante que a função de pausa só inicia quando o evento é sinalizado, permitindo um controle sincronizado das operações de pausa e retoma da bolsa.

O evento "hEventoLer" é utilizado para sincronizar a leitura de mensagens de um pipe do lado do cliente na função "**Recebe"**. Ao ser sinalizado, permite que a função inicie a leitura assíncrona do pipe, garantindo que ocorra apenas quando necessário. Na função "Recebe", após aguardar pelo evento, o código prepara a leitura assíncrona e espera sua conclusão.

Manual de utilização

Bolsa

Comandos	Descrição
addc <nome-empresa> <número-ações> <preço-ação>	Acrescentar uma empresa
listc	Listar todas as empresas
stock <nome-empresa> <preço-ação>	Redefinir o custo das ações de uma empresa
users	Listar utilizadores
pause <número-segundos>	Pausar as operações de compra e venda
close	Encerrar a plataforma

Cliente

Comandos	Descrição
login <username > <password>.	Autenticar um utilizador
listc	Listar todas as empresas
buy <nome-empresa> <número-ações>	Comprar ações
sell <nome-empresa> <número-ações>	Vender ações
balance	Consultar saldo
exit	Sair da aplicação

Requisitos implementados

ID	Descrição funcionalidade/requisito	Estado
1	Operações de compra	Implementado
2	Operação de venda	Implementado
3	Variação de preços/ações	Implementado
4	Ficheiro de leitura de empresas	Implementado
5	Registry	Implementado
6	Named Pipes	Implementado
7	Memória Partilhada	Implementado
8	Ler ficheiro utilizadores	Implementado
9	Comandos Bolsa	Implementado
10	Comandos Cliente	Implementado

Decisões tomadas

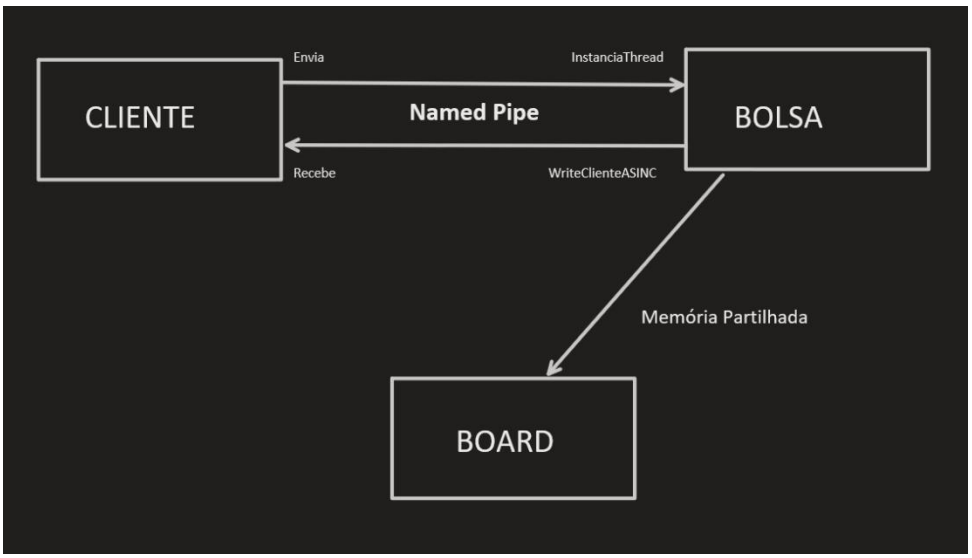
Variância de Preços/Ações:
-Venda de Ações: Quando um utilizador vende ações, o preço da ação diminui. Onde o preço da ação é reduzido por uma quantidade proporcional ao número de ações vendidas e às ações disponíveis no mercado, multiplicado por um fator de 10%.
-Compra de Ações: Quando um utilizador compra ações, o preço da ação aumenta. Aumentando o preço da ação pela mesma fórmula.

Memória Partilhada:
Optamos por criar a memória partilhada em ambos os programas (bolsa e board), para garantir uma maior eficiência e sincronização dos dados.

Threads 'Recebe':
Decidimos criar uma thread 'Recebe' no cliente para receber os dados que são enviados pela bolsa, pois assim conseguimos implementar a funcionalidade da bolsa fechar e consequentemente fechar todos os clientes.

Variáveis globais:
numMaxEmpresas: Variavel na board que vai guardar o número de empresas para mostrar lidas na linha de comandos.
MAX_USERS: Vai guardar o número máximo de utilizadores lidos no registry.

Diagrama com os mecanismos de sincronização e comunicação



Conclusão

O trabalho prático evidencia uma implementação sólida e eficiente de uma bolsa de valores online, usando estruturas de dados adequadas e mecanismos de comunicação como named pipes e memória partilhada. A utilização de mutexes e eventos garante a sincronização e consistência dos dados. Em resumo, o projeto demonstra um domínio sólido dos conceitos abordados e uma aplicação hábil das tecnologias utilizadas.

Este trabalho foi de grande importância para aprofundar o nosso conhecimento.