Sistemas Operativos II
2.ª Meta

João Almas | 2021138417 | P5
Paulo Sá | 2021142819 | P5

Índice

1. Introdução

No âmbito da unidade curricular de Sistemas Operativos II, foi proposto o desenvolvimento de uma aplicação que simula uma bolsa de valores online. Este projeto consiste na implementação de diversos programas que operam em conjunto para criar uma plataforma de transação de ações entre utilizadores. Ao longo deste relatório, serão detalhados os principais aspetos técnicos e funcionais do sistema desenvolvido. Neste relatório, serão abordados detalhes técnicos da implementação, incluindo a arquitetura do sistema, as estruturas de dados utilizadas, os mecanismos de comunicação e de sincronização, bem como outras decisões de design e implementação. O objetivo é fornecer uma visão abrangente do funcionamento do sistema e das soluções adotadas para alcançar seus objetivos.

2. Programas

2.1. Bolsa

A bolsa é constituída por 4 threads, além das threads dos utilizadores que estão conectadas à bolsa. A thread principal tem o objetivo de abrir as threads, criar variáveis, receber o output do gestor da bolsa e, no final, esperar pelas threads terminarem e dar closehandle para finalizar o programa de forma controlada.

A thread para pausar é criada logo desde o início e é sincronizada com o evento. Quando o gestor quiser ativá-la, é definido o evento e é criado um Waitable Timer para não permitir compras nem vendas.

A thread responsável pela criação de FileMapping em seguida faz o MapViewOfFile, cria um mutex(evento board) e um evento(evento board) para a sincronização com as boards. Em seguida, ela espera pelo evento(avisa) criado na thread principal, que serve para avisar que os dados da empresa foram alterados para atualizar a memória compartilhada e enviar um sinal com o evento(evento board) criado anteriormente pela thread para informar as boards sobre a nova informação.

Tem uma thread que serve para atender os clientes usando um semáforo e um evento. Cria uma instância do named pipe em modo não bloqueante e fica à espera no GetOverlappedResult. Em segundo, quando se conecta um novo cliente, cria uma thread para ele e refaz o processo até que o contador atinja 0 e não receba mais clientes.

A thread com os clientes é uma thread básica de pedido-resposta em modo não bloqueante, onde recebe uma mensagem do cliente, lê o ID, depois lê a mensagem, processa-a e responde e envia.

ID	Descrição funcionalidade / requisito	Estado
1	comonicacao do memoria partinhada	implementado
2	Comonicao por pipes	implementado
3	Comandos - addc - listc - stock - users - pause - close	implementado
4	Os utilizadores têm contas, que inicialmente são lidas de um ficheiro de texto	implementado
5	A criação de empresas e ações pode ser lida de um ficheiro de texto	implementado
6	A compras fazem aumentar o preço e as vendas fazem baixar (após a venda/compra), segundo uma lógica natural de procura/oferta.	implementado

Comandos

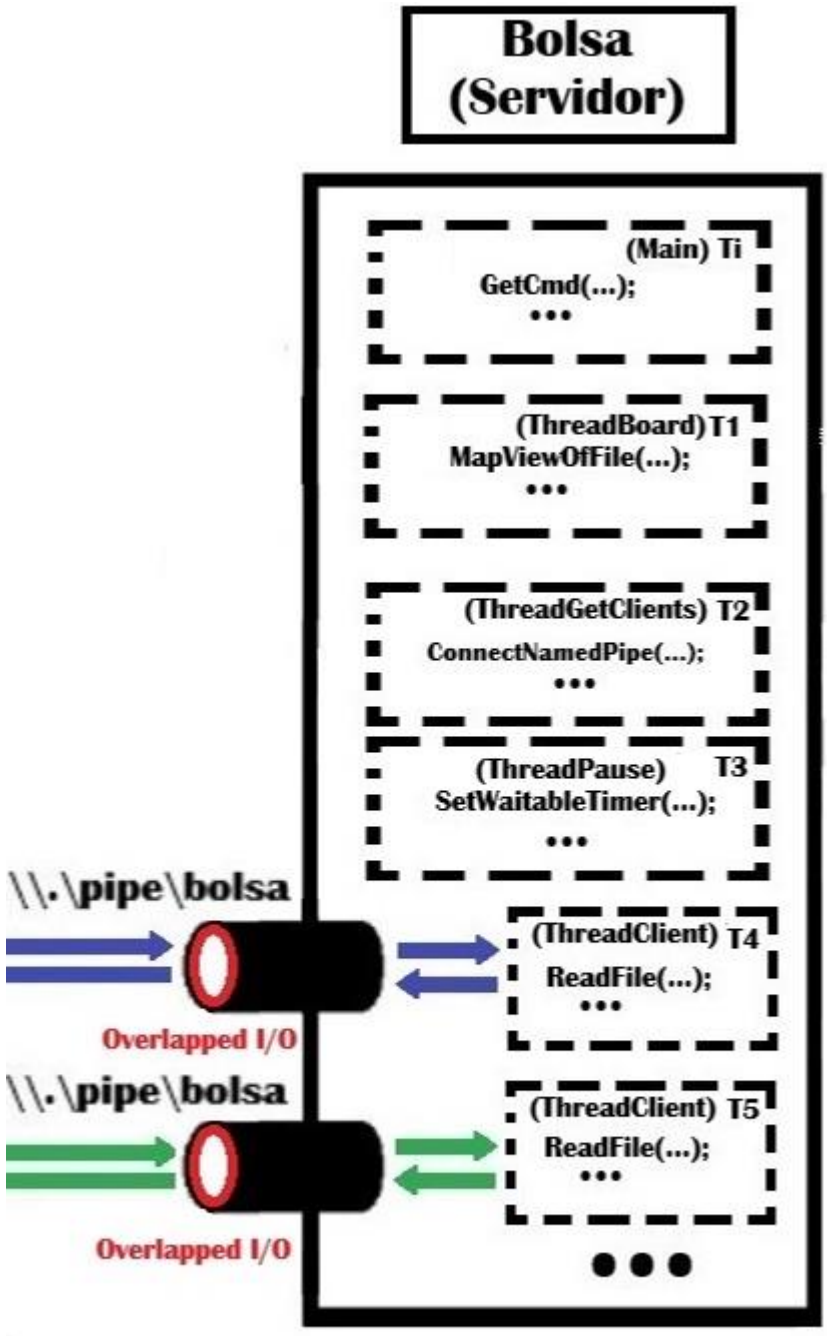
O addc é protegido por Critical Section quando está a utilizar memória partilhada, como outras threads da bolsa, e cria uma empresa, dando set a um evento que avisa a thread da memória partilhada.

O listc e users apenas listam e utilizam Critical Section para não ter problemas ao usar as variáveis .

O close não apanha a fase da Critical Section para si para mudar a flag de continuação e, em seguida, sair do ciclo da bolsa e esperar pelas threads terminarem para finalizar o programa.

O pause tem uma thread só para si que muda uma flag para true para não permitir comprar ou vender, também utilizando Critical Section.

O stock muda o valor da empresa e dá set a um evento que avisa a thread da memória partilhada.


Bolsa
(Servidor)
(Main) Ti
GetCmd(...);
(ThreadBoard) T1
MapViewOfFile(...);
(ThreadGetClients) T2
ConnectNamedPipe(...);
(ThreadPause) T3
SetWaitableTimer(...);
\\.\pipe\bolsa
(ThreadClient) T4
ReadFile(...);
Overlapped I/O
\\.\pipe\bolsa
(ThreadClient) T5
ReadFile(...);
Overlapped I/O

2.2. Board e a boardGUI

Esses programas têm base em arquivos mapeados com a ajuda do Win32. Foi usada a função OpenFileMapping em modo FILE_MAP_READ. Em seguida, os boards fazem MapViewOfFile para criar uma visão. Agora, a board tem acesso à memória compartilhada da bolsa. Para sincronizar a informação, usamos um mutex é um evento. O mutex protege a memória se está sendo acessada por mais de um processo ao mesmo tempo, e o evento dispara quando há novos dados, para evitar uma espera ativa.

Para sair no board de linha de comando, foi usada uma thread que espera que o utilizador clique em uma tecla. Já na boardGUI, foi utilizada a interface gráfica da Microsoft Win32 para mostrar as informações e permitir interação com o programa. Ela abre com dados padrão, mas que podem ser alterados, como o eixo do X (valor máximo do gráfico) e do Y (número de empresas que aparecerão no gráfico).

No menu da GUI, existem quatro opções: uma para ver as informações dos criadores, onde aparece um dialog box com as informações e um botão para sair; outra para mudar o valor do eixo do X, também com um dialog box com uma frase indicando o que é, e uma caixa de texto que só aceita números, com botões para OK e cancelar; outra para mudar o valor do eixo do Y, com as mesmas opções da alteração do eixo do X; e outra para sair. Todas são modais, capturando o foco de interação até serem fechadas, e foram feitas utilizando os recursos disponibilizados pelo Visual Studio.

No topo da tela, no meio, aparece a empresa que foi comprada por último, e mais abaixo, também no meio, há um gráfico que mostra o valor da empresa mais valiosa no momento. Para threads que estão interagindo com a memória compartilhada, é enviado um evento para repintar a tela.

Board

ID	Descrição funcionalidade / requisito	Estado
1	Número de empresa especificado através de um argumento da linha de comandos	implementado
2	A comunicação via memória partilhada.	implementado

BoardGUI

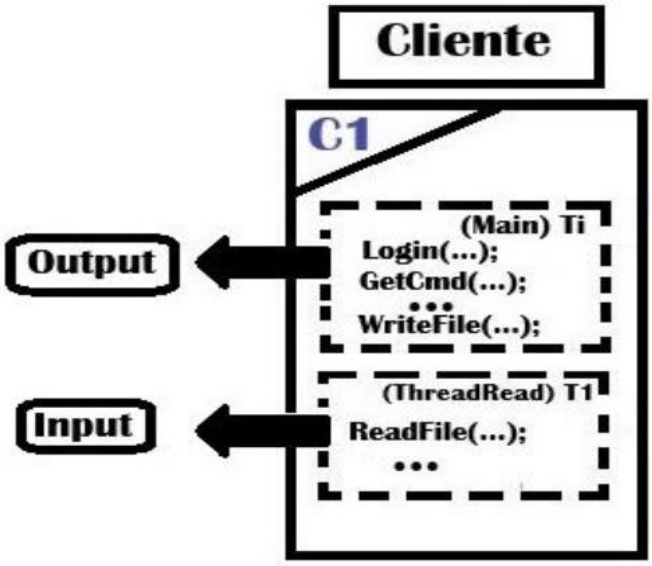
ID	Descrição funcionalidade / requisito	Estado
1	Número de empresa especificado através da interface gráfica / valor máximo do gráfico	implementado
2	100% interfaccia grafica	implementado
3	A comunicação via memória partilhada	implementado

2.3. Cliente

O programa cliente tem um funcionamento simples, com duas threads dedicadas a enviar mensagens para a bolsa e receber mensagens dela. As threads podem operar de forma independente. Inicialmente, um pipe em modo bytes é aberto para permitir a comunicação. Quando o utilizador faz um pedido, uma das threads envia mensagens para a bolsa, estruturando a mensagem com um ID e os dados correspondentes, utilizando um WriteFile bloqueante, já que é enviada imediatamente. Retornando em um ciclo, aguardando a interação do utilizador.

A thread de recebimento de mensagens permanece em espera, utilizando um ReadFile não bloqueante, e bloqueia em um GetOverlappedResult à espera de mensagens da bolsa. Ela lê um DWORD para identificar o tipo de mensagem enviada pela bolsa e, em seguida, realiza outra operação de leitura com ReadFile para processar e mostrar a mensagem ao utilizador.

ID	Descrição funcionalidade / requisito	Estado
1	Comandos - buy - sell - balance - exit	implementado
2	Bolsa – cliente à named pipes	implementado
3	envio e recebimento de msg são independentes	implementado



3. Comunicação entre Processos

3.1. Cliente ⇔ Bolsa

A comunicação entre o Cliente e a Bolsa foi realizada através de named pipes. No lado da Bolsa, uma thread esperava a conexão de um cliente e, uma vez estabelecida a conexão, uma nova thread era lançada para lidar com esse cliente. No lado do cliente, uma thread recebia a saída do servidor, enquanto outra enviava solicitações para o servidor.

3.2. Bolsa ⇔ Board

A comunicação entre a Bolsa e a Board foi feita através de memória compartilhada. O programa Bolsa inicializou e gerenciou a memória compartilhada, criando um arquivo para ela e estabelecendo um mapeamento para uma visualização que corresponde ao tamanho exato de uma estrutura "empresa". Do lado da Board, uma visualização foi criada para acessar as informações na memória compartilhada. Um evento era acionado pela Bolsa sempre que as informações na memória compartilhada eram atualizadas, permitindo que o processo da Board soubesse quando recolher informações. Para garantir a integridade dos dados, um mecanismo de exclusão múltipla (mutex) foi utilizado para proteger o acesso à memória compartilhada.

4. Bibliotecas

Foi ponderada a possibilidade de utilizar bibliotecas estáticas para aproveitar funções e estruturas nos programas, visando a organização e reutilização de código. No entanto, devido à pressão do tempo e às demandas urgentes do projeto, decidimos não investir na implementação de DLLs. Essa escolha foi feita considerando a priorização de outras tarefas críticas e a necessidade de entregar o projeto dentro do prazo estabelecido.