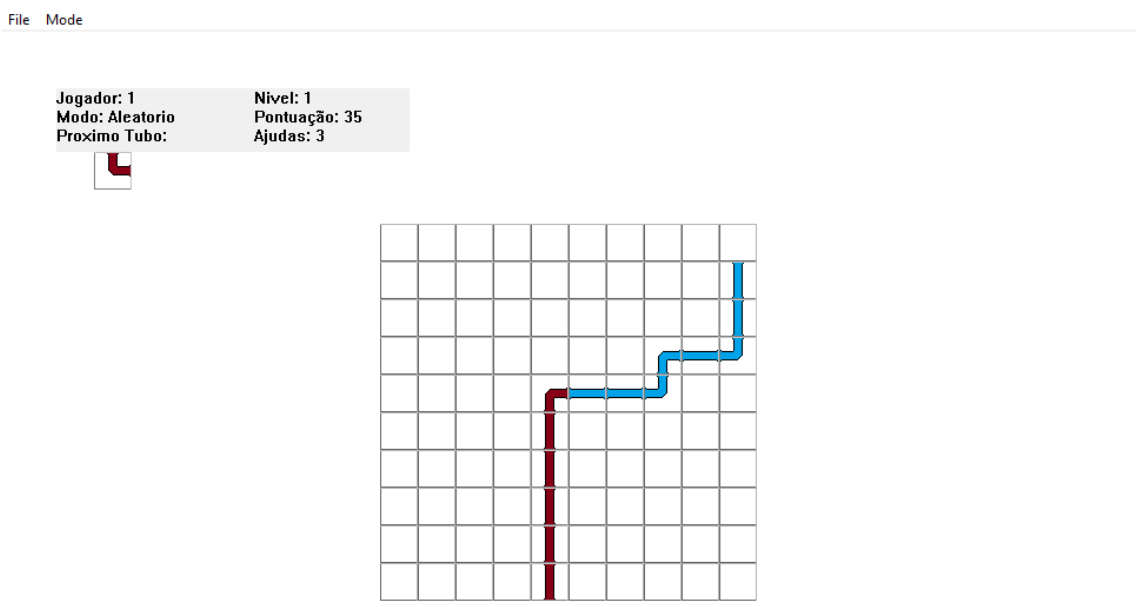
Sistemas Operativos II

Trabalho Prático – Aplicação de Apoio ao Processo de Gestão de Projetos e Estágios.

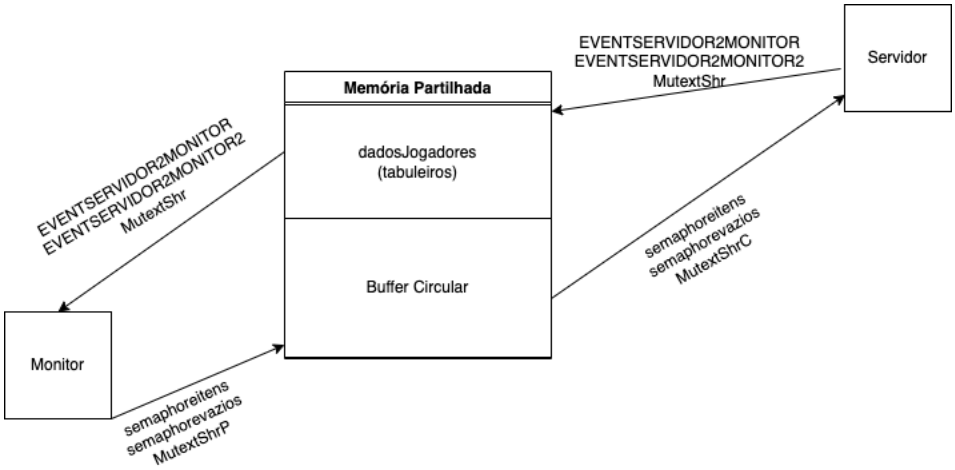


Daniel Fernandes (LEI-PL) - a2020116565@isec.pt
Hugo Jorge (LEI-PL) - a2020116988@isec.pt

Memória Partilhada

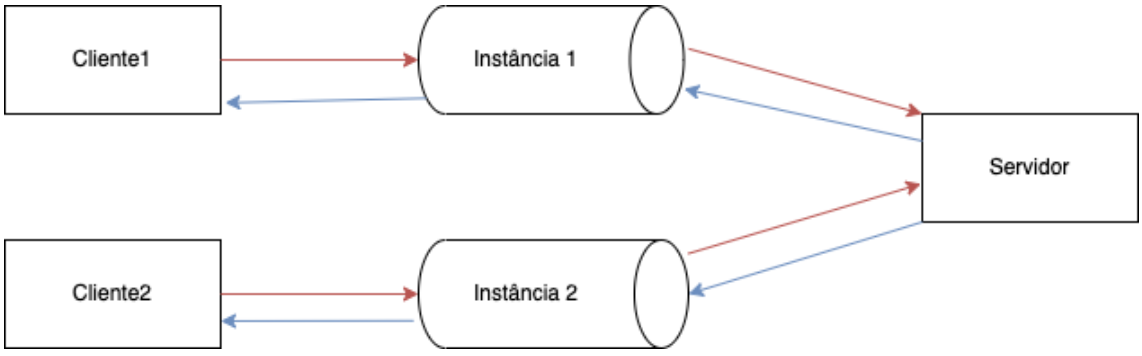
O servidor comunica com o monitor e o monitor com o servidor a partir de memória partilhada. A comunicação entre servidor e monitor é através da estrutura dadosJogo que contém os tabuleiros dos jogadores e dados referentes ao tabuleiro e estado do jogo.

A comunicação entre monitor e servidor é um buffer circular. Consideramos o tamanho do buffer de 10 mensagens, pois achamos suficiente em relação ao número de comandos que o monitor pode executar.



NamedPipes

A comunicação entre clientes e o servidor e vice-versa é realizada exclusivamente a partir de namedpipes. É criado um namedpipe no servidor preparado para overlapp. Para a realização da escrita e leitura do namepipe funcionar perfeitamente e apenas usarmos até 2 instância do mesmo, usamos os pipes com overlapp.



Estruturas Comuns ao Servidor e Monitor - Servidor_Monitor.h

A estrutura dadosJogo é a memoria partilhada em que se encontra os dados relativos aos tabuleiros dos jogadores a partir de uma estrutura tabuleiros e umas variáveis inteiras que relatam o número de linhas, colunas, tempo da água, estado do jogo e número de jogadores.

A estrutura bufferCircular é a estrutura correspondente ao buffer circular. Esta é composta por dois inteiros correspondentes à posição de escrita e de leitura no buffer e um array de estruturas do tipo MSGbufferCircular que tens campos específicos para as mensagens como um array de caracteres e 3 inteiros.

Estruturas do Servidor – Servidor.h

A estrutura dadosThreadJogador é uma estrutura que contém os dados importantes para a realização do jogo como as posições iniciais e finais dos tabuleiros, variável de sinalização de termino, número de jogador, ponteiro para a memoria partilhada relativa aos tabuleiros, e Handles para mutex de acesso à memória partilhada, de acesso a dados desta mesma estrutura e de acesso à variável de sinalização de termino. Tem também Handles para o evento a sinalizar quando existe dados novos na memória partilhada, para a thread e waitable timer correspondente da água e para um evento do qual a thread de jogar ficará à espera. Existirão 2 estruturas destas. Uma para cada jogador.

A estrutura dadosComandos contem os handles para os semáforos e mutex do buffer circular, um ponteiro para o buffer circular e um ponteiro para as estruturas das threads dos jogadores.

Estruturas do Monitor – Monitor.h

A estrutura dadosMonitor contem um ponteiro para a memória partilhada dos tabuleiros, um ponteiro para uma flag de termino das threads, um inteiro do jogador que a thread está a ler e handles para três mutex e um evento. Os mutex para acesso à flag de sair, acesso à memória partilhada e acesso ao stdout e o evento para saber quando foi colocado nova informação na memória partilhada de um jogador em específico. Existe 2 estruturas deste tipo para cada thread que imprimirá informação relativa a cada jogador.

A estrutura dadosComandos contem os handles para os semáforos e mutex do buffer circular e um ponteiro para o buffer circular. Contem também um ponteiro para os dadosMonitor para poder ter acesso aos mutex aí guardados, para confirmações de introdução de comandos corretos.

Estruturas do Cliente – Cliente.h

O cliente tem duas estruturas que são as estruturas DADOSBM e estrutura dadosThreadCli. A estrutura DADOSBM serve para armazenar os handles para os bitmaps e os hdc dos mesmos. Serve também para armazenar informações relativas aos bitmaps como o seu tamanho. A estrutura dadosThreadCli é a estrutura enviada para a thread que contem uma estrutura DADOSBM, o handle para o namedpipe , para um mutex de acesso aos dados, para a janela principal e para uma janela de texto. Contem também uma enumeração que corresponde ao estado do jogo, as posições iniciais onde vai ser impresso o tabuleiro, uma flag para terminar e handles para a thread de sinal de vida e para o waitable timer do sinal de vida. Contem também uma estrutura do tipo dadosServ2Cli, que corresponde à estrutura de comunicação de Servidor para Cliente.

Estruturas Comuns ao Servidor e Cliente – Servidor_Cliente.h

As estruturas comuns ao servidor e cliente são as estruturas de comunicação. Essas estruturas são diferentes conforme o sentido da mensagem. Para a comunicação de cliente para servidor é utilizado a estrutura dadosCli2Serv. Esta estrutura contém 3 inteiros dos quais dois deles servem para enviar as coordenadas das peças em que o cliente clica e outro é usado em alguns momentos para auxiliar a informação enviada. Contem também uma enumeração com chaves do desenvolvimento do jogo. Esta enumeração serve para identificar o tipo de mensagem enviado.

Esta enumeração é enviada em ambos os sentidos, ou seja, está presente na estrutura usada no sentido de Servidor para cliente. Essa estrutura é a dadosServ2Cli. Esta mesma contém o id do jogador, uma estrutura do tipo dadosJogo onde contém informações do jogo, a ultima posição da agua e os tabuleiros e também a enumeração referida anteriormente.

Mecanismos de sincronização

O monitor é alertado que existe informação na memória partilhada através de 2 eventos. Um evento de cada jogador. Para mostrar o evento existirá 1 ou 2 threads conforme a quantidade de jogadores. Para não haver acessos em simultâneo à memória partilhada é usado um mutex.

O buffer circular tem 2 semáforos cujo nomes são conhecidos pelo servidor e monitor e 2 mutex. Um para os produtores (monitores) e um para o consumidor (servidor).

O servidor e monitor conhecem os nomes de todos estes mecanismos de sincronização referidos anteriormente.

Para que cada monitor consiga imprimir os mapas de jogo e escrever os comandos, estes são feitos numa thread à parte.

No servidor existe um evento que serve para as threads dos jogadores estarem paradas enquanto a água corre. Isto é apenas para permitir que enquanto o jogo decorre se possa executar os comandos do servidor.

Existe em ambos um mutex para o acesso à variável de termino de cada programa e um mutex para acesso às variáveis das estruturas do tipo dadosThreadJogador do servidor.

Também existem dois waitable timers que irão servir para a colocação da água nos tubos.

Manual de utilização.

Deve ser executado primeiro o servidor e depois os monitores se pretendido.

De seguida deve se iniciar um cliente e no menu carregar em iniciar jogo. O programa perguntará ao utilizar o nome, o namepipe. Caso o utilizador não digite 15 caracteres, tamanho de um ip, o jogo assume que o servidor está a correr na mesma máquina. De seguida o programa pergunta ao utilizador quantos jogadores são. Se for 1 o jogo arranca de imediato, caso seja dois espera pelo próximo cliente inicie. Assim que estiver os jogadores devidos conectados o servidor lança uma thread que faz a gestão das threads do jogador.

Cada uma destas threads prepara uma thread e um waitable timer. Esta última thread espera que o seu waitable timer desperte e vai colocar água no jogo caso seja possível. Caso a água chegue a um tubo que não permita a sua continuação ou uma parede, muda o valor a uma variável a sinalizar que perdeu o jogo.

Apenas são aceites os comandos de suspender se o jogo estiver a decorrer e de retomar caso o jogo esteja suspenso, pois caso contrário não teria sentido.

Os comandos são impressos no ecrã para que o utilizador não tenha dúvidas sobre eles.

No monitor pode digitar os comandos que pretender que apenas são enviados ao servidor os que tiverem uma sintaxe válida. Também estes são impressos no ecrã.

Do lado do Servidor são também impressos os comandos permitidos e a sua descrição.

O jogador cada vez que clica no tabuleiro é enviado um pedido ao jogador com as posições da peça em questão.

O tabuleiro apenas é atualizado quando o servidor diz que a jogada é valida.

Implementação dos Comandos do Servidor

Ao introduzir o comando de suspender, o servidor faz resume das threads dos jogadores e das threads da água e envia para os clientes uma mensagem a avisar que o jogo está suspenso. Muda também o estado de jogo para suspenso. O comando apenas é realizado se o jogo se encontrar a decorrer.

O comando de retomar apenas é executado quando existe um jogo a decorrer. Quando é executado o servidor resume as threads e volta a preparar o waitable timer das threads da água. O servidor também envia uma mensagem para os Clientes a avisar que é para continuar o jogo.

O comando encerrar cancela os waitables timers e mete a flag de termino a 1. Também manda evento para todas as threads e mete na memoria partilhada uma mensagem que indica ao monitor para terminar e manda uma mensagem para o cliente para terminar. Este recebe e simula um WM_CLOSE do qual avisa o jogador com uma dialogbox que o servidor mandou encerrar.

Por fim o comando listar mostra a quantidade de jogadores e a pontuação no preciso momento.

Implementação dos Comandos do Monitor

No monitor os comandos apenas são enviados para o servidor se o comando for válido em termos de estrutura.

O comando parar água envia para o servidor uma mensagem que faz com que o servidor mude o waitable timer para que a próxima execução demore a quantidade de segundos da mensagem.

O comando insere bloco envia para o servidor a mensagem e o servidor coloca no tabuleiro um bloco no tabuleiro do jogador e nas coordenadas enviadas.

O comando ativa aleatório envia para o servidor apenas o comando e o servidor muda para o modo de jogo para aleatório.

O comando desativa aleatório faz algo similar ao anterior só que o servidor muda o modo de jogo para sequencial.

Implementação das jogados do Cliente

Quando o jogador clica no rato esquerdo ou direito, o programa cliente deteta se clicou na área do mapa de jogo e envia uma mensagem ao servidor com os índices da peça clicada e com o tipo de jogada.

O servidor valida a jogada e se for valida atualiza o mapa. O servidor envia para o cliente uma mensagem a dizer se a jogada foi valida ou não.

Para o parar a água, quando o jogo está ativo e o rato move-se é preparado um trackmouseevent que lançará um WM_MOUSEHOVER quando o rato estiver parado numa célula do tabuleiro durante 2 segundos. Ao receber este evento o cliente verifica se foi a última posição de água e envia para o servidor este pedido. Após a água parar, é detetado quando o cursor sai da célula em questão e envia mensagem ao servidor para a água retornar. O servidor apenas permite parar a água 3 vezes. A partir dai ignora as mensagens recebidas deste género.

Decisões Tomadas

Decidimos que apenas detetávamos o parar a água por pedido do cliente com o rato parado em cima da célula, porque refletir um mouse over em que o rato podia estar a mover-se dentro da mesma célula iria necessitar de cálculos constantes sempre que o rato se movia. Portanto não vimos necessidade do mesmo.

Implementações

ID	Descrição funcionalidade / requisito	Estado
001	Obragatoriedade de o servidor ser o primeiro processo a ser lançado	implementado
002	Só pode haver uma instância do servidor em funcionamento	implementado
003	Dimensões da área de jogo e tempo da água, armazenadas no registry ou passadas por parametro ao executar	implementado
004	Origem e fim da água determinados de maneira aleatória e em posições do tabuleiro diagonalmente opostas	implementado
005	Processamento dos comandos do monitor feito pelo servidor	implementado
006	Processamento das jogadas provenientes dos clientes e atualização do mapa de jogo	implementado
007	Comandos do servidor de listar jogadores e respetiva pontuação, suspender e retomar o jogo e encerrar	implementado
008	Verificação do sinal de vida dos clientes por parte do Servidor	implementado
009	Possibilidade de existirem mais instâncias funcionais do monitor	implementado
010	Possibilidade do Monitor parrar a água de um jogador em específico durante um tempo definido	implementado
011	Possibilidade do Monitor inserir um bloco em uma posição do tabuleiro dum jogador em específico	implementado
012	Possibilidade do Monitor ativar/desativar o modo aleatório para a sequência de peças/tubos	implementado
013	Modo de jogo individual e multijogador	implementado
014	Modo de jogo individual com níveis de dificuldade que vão incrementando	implementado
015	Comunicação entre o servidor e os clientes feita exclusivamente por named pipes, em ambas as direções.	implementado
016	Clientes apenas comunicam com o Servidor	implementado
017	Servidor comunica com os monitores através de memória partilhada	implementado
018	Monitores comunicam apenas com o Servidor e através de buffer circular	implementado
019	Double buffering aplicado no cliente	implementado
020	Monitores com informação constante do mapa dos jogadores em tempo real.	implementado
021	No cliente, cada clique com o botão esquerdo do rato em cima de uma célula onde ainda não passou a água permite mudar a peça/tubo de forma circular	implementado
022	No cliente, cada clique com o botão direito do rato permite limpar a célula.	implementado
023	Mouseover sobre a célula onde a água se encontra atualmente suspende temporariamente o seu fluxo, até que o cursor saia da célula em causa.	implementado

024	Mouseover só é ativado 2 segundos após o cursor se encontrar sobre a célula.	implementado
025	Mouseover só acontece no máximo 3 vezes por jogo.	implementado
026	2 conjuntos de bitmaps que podem ser alternados em qualquer momento do jogo	implementado
027	Uso de dll	implementado
028	Menu inicial	implementado
029	Menu para modo de bitmaps	implementado
030	Popups para informação relativa ao funcionamento jogo	implementado
031	Verificação de saída da aplicação	implementado
032	Impressão em tempo real do tabuleiro	implementado
033	Espera pelo Jogador 2 no modo multijogador	implementado