Instituto Superior de Engenharia

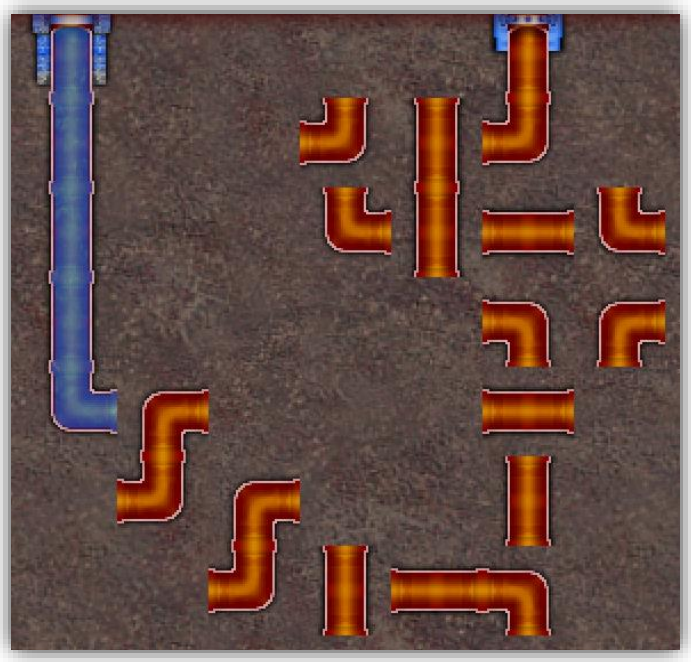
Politécnico de Coimbra

Licenciatura em Engenharia Informática

Ramo de Desenvolvimento de Aplicações

Sistemas Operativos II 2021/2022

Jogo dos Tubos



Trabalho realizado por:

Beatriz Pinto | 2020144095

Eduardo Correia | 2020139576

Índice

Mecanismos de comunicação e sincronização

A comunicação entre o servidor e o monitor é feita exclusivamente por memória partilhada, ou seja a verificação do conteúdo das células e a sua atualização é feita de forma direta na memória partilhada, enquanto a restante comunicação do monitor para o servidor é feita através do paradigma de produtor / consumidor(buffer circular).

```
void criaFileMap(HANDLE* hFileMap, BOOL* primeiroProcesso, TCHAR* caminho);
```

- Função responsável por criar o File Map que servirá para o servidor e o monitor se comunicarem pela memória partilhada.

- Servidor

```
void gereComandos(DadosThreads* dados, Comandos* cel);
```

- Função que recebendo o comando que foi passado pelo monitor está responsável por executá-lo.

```
DWORD WINAPI ThreadAgua(LPVOID param);
```

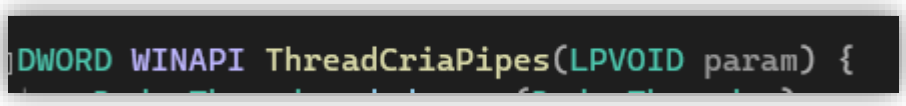
- Thread que implementa o mecanismo da água que se encontra sempre atualizado na memória partilhada.

```
DWORD WINAPI ThreadServidor(LPVOID param);
```

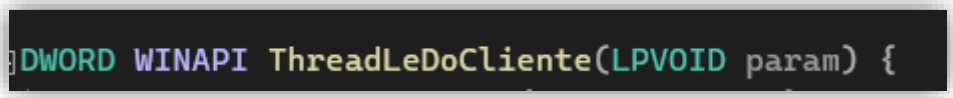
- Thread responsável pela comunicação Monitor/Servidor através do buffer circular, do lado do servidor que recebe os comandos enviados pelo lado do monitor e manda executá-los à função gereComandos().

```
DWORD WINAPI ThreadTeclado(LPVOID param);
```

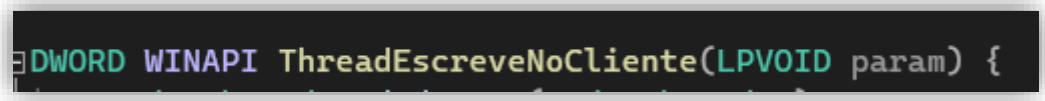
- Thread que fica à espera de comandos inseridos pelo próprio servidor e os executa.



-Thread que é responsável por criar os named pipes e esperar que um cliente se conecte.

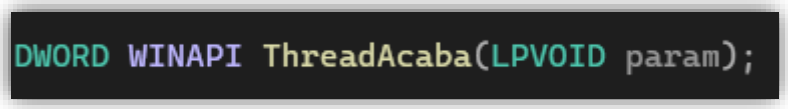


-Thread que recebe e processa a informação dos respetivos clientes.

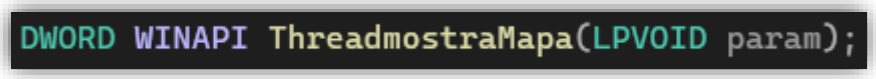


-Thread responsável por mandar informações para o cliente, servindo para mantê-lo atualizado sobre o seu mapa de jogo.

- Monitor



- Thread que fica responsável por aguardar que o servidor termine e terminar também o monitor.

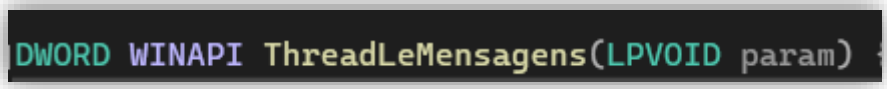


- Thread que tem como função mostrar o mapa e o decorrer da água no ecrã de segundo em segundo.

- Thread responsável pela comunicação Monitor/Servidor através do buffer circular, do lado do monitor que envia os comandos inseridos por quem está a controlar o monitor para o servidor para que possam ser executados.

- Cliente

- Thread que fica responsável por aguardar que o servidor termine e terminar também os clientes.



- Thread que fica responsável por ler a informação enviada pelo servidor.

- Thread que fica responsável por ir atualizando o mapa de jogo.

- Função associada à criação da janela que processa os eventos por esta despoletados.

Estruturas de dados

```
typedef struct {
    TCHAR tubos;
    BOOL agua;
}CelulaBuffer;
```

- O mapa de jogo é formado por células desta estrutura pois estas podem conter tubos ou água.

```
typedef struct {
    int tamMap;
    int tempAgua;
    int paraAgua;
    DWORD inicioX;
    DWORD inicioY;
    DWORD destinoX;
    DWORD destinoY;
    BOOL aleatorio;
    CelulaBuffer mapa[TAM_BUFFER][TAM_BUFFER];
}AreaDeJogo;
```

- Esta estrutura é responsável por guardar as informações acerca da área de jogo.

```
typedef struct {
    DWORD pararAgua[2];
    BOOL modoAleatorio;
    DWORD inserirParedes[3];
}Comandos;
```

- Esta estrutura é responsável por guardar e passar comandos do monitor para o servidor.

```c
typedef struct {
    int nMonitores;
    int posE; //proxima posicao de escrita
    int posL; //proxima posicao de leitura
    AreaDeJogo mapaJogo;
    Comandos comandos[TAM_BUFFER];
}BufferCircular;
```

- Estrutura na qual é feita a comunicação em buffer circular entre o monitor e o servidor.

```c
typedef struct {
    //DWORD totalClicks;
    TCHAR nome[100];
    HANDLE pipe;
    TCHAR tipoJogo[100];
    DWORD hover;
    DWORD clickTipo;
    DWORD linhaColuna[2];
    TCHAR ordem[6];
    DWORD nivel;
    DWORD clis;
    TCHAR pecas[13];
    AreaDeJogo mapaJogo;
    HANDLE iniciaJogo;
    BOOL aleatorio;
}MensagesCliente;
```

- Estrutura que permite armazenar informações necessárias para que o servidor e os clientes consigam trocar dados entre si.

```c
// estrutura do named pipe
typedef struct {
    HANDLE hPipe; // handle do pipe
    OVERLAPPED overlap;
    BOOL activo; //representa se a instancia do named pipe esta ou nao ativa, se ja tem um cliente ou nao
} PipeDados;
```

- Estrutura que diz respeito aos named pipes que são usados para fazer a comunicação entre o servidor e os clientes.

```c
//estrutura de apoio
typedef struct {
    BufferCircular* memPar; //ponteiro para a memoria partilhada
    MensagesCliente cli[2];
    HANDLE hSemEscrita; //handle para o semaforo que controla as escritas (controla quantas posicoes estao vazias)
    HANDLE hSemLeitura; //handle para o semaforo que controla as leituras (controla quantas posicoes estao preenchidas)
    HANDLE hMutex;
    PipeDados hPipes[2];
    HANDLE hEvents[2];
    HANDLE hPipeCli;
    HANDLE acabou;
    HANDLE comecaCliente;
    HANDLE inicia;
    HANDLE clienteEscreveu;
    HANDLE clienteIniciou2;
    HANDLE clientePipe;
    HANDLE atualizaJogo;
    HANDLE leDoCliente;
    HWND hWnd;
    int nClientes;
    int terminar; // 1 para sair, 0 em caso contrário
}DadosThreads;
```

- Estrutura que permite guardar os mecanismos de sincronização e comunicação que irão auxiliar durante o decorrer do jogo.

Requisitos implementados

Descrição funcionalidade / requisito	Implementado	Não implementado	Implementada Parcialmente
Regitry	X		
Servidor	X		
Jogo Individual	X		
Jogo Multijogador			X
Monitor	X		
Cliente	X		
Sinal de vida			X
Níveis	X		
BitMaps	X		
Comandos	X		
Named pipes	X		
Memória Partilhada	X		

Manual de utilização

- Servidor

O servidor ao ser lançado pode lhe ser passado na linha de comandos o tamanho do mapa e o tempo que a água demorara a começar a correr. Caso não lhe sejam passadas estas informações o servidor vai buscar as informações que estão guardadas no registry.

Após ser lançada uma instância do servidor este irá criar o ficheiro para a memória partilhada para se poder comunicar com o monitor, assim como os named pipes necessários à comunicação com o cliente, aguardando que algum cliente se conecte e inicie o seu jogo, despoletando todas as funções necessárias ao funcionamento do jogo.

- Monitor

Ao lançar o monitor este irá assegura-se de que o servidor já está a correr, caso contrário não será possível não é possível iniciar o monitor.

Depois de iniciado um monitor e assim que o jogo seja iniciado o monitor irá começar a mostrar o mapa com a informação atualizada. No decorrer o jogo o monitor pode receber comandos que irão ser enviados para o servidor para que este os execute.

Posteriormente, assim que o servidor termine, o monitor também será terminado.

- Cliente

Ao lançar o cliente este irá assegura-se de que o servidor já está a correr, caso contrário não será possível não é possível iniciar o cliente.

O cliente por ser em modo gráfico irão lhe aparecer um campo onde ele irá colocar o seu nome e dois botões que dizem respeito ao tipo de jogo a escolher. Também é disponibilizado um menu que lhe permite alternar entre 2 conjuntos de BitMaps. Caso o cliente escolha o tipo de jogo: individual, irá aparecer-lhe o mapa de jogo e a ordem das peças que este tem disponível para jogar. Ao ganhar este passará para o próximo nível, onde a água corre mais rápido. Se não conseguir fazer com que a água chegue ao destino o cliente irá perder e vai ser notificado.