

Departamento de Engenharia Informática e de Sistemas

Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática

Sistemas Operativos 2 - 2021/2022

Relatório do Trabalho Prático:

1ª meta

Turma Prática Nº 1

Francisco Simões | a2019133920@isec.pt

Ricardo Ferreira | a2016020798@isec.pt

Índice

Conteúdo

Mecanismos de Comunicação Sincronização (Servidor)	3
Mecanismos de Comunicação Sincronização (Monitor)	4
Estruturas de dados	5
Named Pipes	7
Manual de Instruções.....	9

Mecanismos de Comunicação

Sincronização (Servidor)

```
BOOL initMemAndSync(DadosSync* cdata, DWORD mapSize, DWORD waterTime)
```

A função **initMemAndSync** inicializa os mecanismos de comunicação e de sincronização.

```
+DWORD WINAPI ThreadEsperaMonitor(LPVOID param) { ... }  
+DWORD WINAPI ThreadCicloAgua(LPVOID param) { ... }  
+DWORD WINAPI ThreadConsumidor(LPVOID param) { ... }
```

A thread **ThreadEsperaMonitor** aguarda o release do semáforo **hSemMonitor** para que o servidor somente comece o jogo quando o monitor está instanciado.

A thread **ThreadCicloAgua** contém a lógica do jogo implementada e a inserção da informação atualizada na memória partilhada.

A thread **ThreadConsumidor** corresponde ao consumo de informação recebido do **Monitor** (Produtor), recebe sempre um comando, o qual é interpretado depois no **executaComando()**;

Mecanismos de Comunicação

Sincronização (Monitor)

```
BOOL initMemAndSync(DadosSync* cdata)
```

Tal como no Servidor, a função **initMemAndSync** inicializa os mecanismos de comunicação e de sincronização.

```
DWORD WINAPI ThreadMonitorLeitura(LPVOID dados) { ... }  
DWORD WINAPI ThreadEncerraMonitor(LPVOID param) { ... }  
DWORD WINAPI ThreadProdutor(LPVOID param) { ... }
```

A thread **ThreadMonitorLeitura** espera por um evento **hEventEscreve**, que corresponde ao momento que o mapa é atualizado no servidor, para depois aceder à memória partilhada (sharedMem->mapa) por exclusão mútua.

A thread **ThreadEncerraMonitor** espera pelo **release** do semáforo do servidor **hSemLock**, que acontece quando o servidor termina, o que obriga o término do monitor.

A thread **ThreadProdutor** espera pelo semáforo de escrita **hSemEscrita** para poder escrever no **buffer circular** (paradigma que usa memória partilhada como meio de comunicação) e dá **release** de um semáforo de leitura **hSemLeitura** para que depois o servidor consiga consumir essa informação, sem que se perca.

Estruturas de dados

```
typedef struct {  
    HANDLE hMapFile; // handle mapeamento  
    SharedMemory* sharedMem; //ponteiro para a memoria partilhada  
    HANDLE hMutex;  
    int terminar; // 1 para sair, 0 em caso contrário (trinco)  
    HANDLE hSem[3]; // 3 semaforos  
    HANDLE hEvent[4]; // 4 eventos  
    ThreadData td;  
    HANDLE hThread[4]; // 4 threads  
}DadosSync;
```

```
typedef struct {  
    int nMonitores; //nProdutores  
    int posR; //proxima posicao de leitura  
    int posW; //proxima posicao de escrita  
    BufferCircular buffer[2]; //buffer circular (array de estruturas)  
  
    BOOL correAgua; // se tem água a correr ou não  
    BOOL modoPecaAleatorio; // se as peças surgem de modo aleatório ou não  
    Mapa mapa; // servidor->monitor  
}SharedMemory;
```

```
typedef struct {  
    int x, y;  
    TCHAR comando[MAXBUFFER];  
}BufferCircular;
```

A estrutura DadosSync é a que controla os dados entre servidor e monitor.

A estrutura SharedMemory contém tudo o que pertence à memória partilhada (mapa para servidor->monitor, sendo o resto respetivo ao paradigma do Buffer Circular).

```

typedef struct {
    // ponto origem
    DWORD origemX;
    DWORD origemY;
    // ponto Destino
    DWORD destinoX;
    DWORD destinoY;

    DWORD mapSize; // mapSize dada uma instância
    DWORD waterTime; // tempo de começo de fluxo de água

    BOOL modoAleatorio; // se on ou off
    TCHAR arrMapa[MAX_MAP_SIZE][MAX_MAP_SIZE]; // mapa em si

    Posicao p; // posicao water
} Mapa;

```

```

typedef struct {
    int atual[2];
    int prox[2];
    int count; //
    int direcao;
} Posicao;

```

A estrutura mapa corresponde aos dados do jogo, sendo que contém uma estrutura membro associada Posicao p, para armazenar os dados das posições dos tubos em conjunto com a direção da água.

Named Pipes

```
typedef struct {  
    HANDLE hEventos[2];  
    PipeData hPipes[2];  
    HANDLE hMutexData;  
    int terminar;  
    Client dadosCliente;  
} ThreadData;
```

```
typedef struct {  
    Mapa mapa;  
    TCHAR nome[100];  
    DWORD modo;  
    DWORD xPos, yPos;  
    DWORD estadoJogo;  
} Client;
```

```
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  
} PipeData;
```

A estrutura ThreadData corresponde à estrutura correspondente pelo controlo dos named Pipes, sendo composta por duas estruturas membros “**hPipes**” (composto pelo pipe, pela estrutura de overlap e por uma flag para saber se o pipe está a ser usado ou não) e “**dadosCliente**” (composto pelos dados do jogo relativos ao cliente).

A estrutura de “**overlap**” tem como funcionalidade tornar os pipes não bloqueantes permitindo assim uma comunicação assíncrona entre o cliente e o servidor.

Interface Gráfica



A técnica de double buffering não foi bem conseguida, daí o mapa cintilar.

Manual de Instruções

Instância Servidor:

- “Iniciar” para carregar a solução final.

Instância Monitor:

- “Bloco” - Inserir blocos Intransponíveis;
- “Paraagua x” - Fecha torneira por x segundos;
- “ModoAleatorio y” - Sequência de peças aleatórias (y - representa on/off);

Instância Cliente:

- “Resume/Pause” – Resumir ou pausar o jogo;
- “Encerra” – Encerrar o Servidor;

Servidor			
Funcionalidades	Implementada	Implementada Parcialmente	Não Implementada
Controla a informação do mapa de jogo (registry)	X		
Determina de forma aleatória a posição da origem e destino da água		X	
Recebe comandos do monitor e desencadeia as ações necessárias	X		
Aceita os jogadores que se ligam através do programa cliente	X		
Recebe por parte dos clientes, as jogadas que pretendem efetuar	X		
Listar os jogadores e respetiva pontuação			X

Suspender e retomar o jogo	X		
Encerrar todo o sistema	X		

Monitor			
Funcionalidades	Implementada	Implementada Parcialmente	Não Implementada
Parar a água durante um período de tempo (especificado em segundos)	X		
Inserir blocos que representam paredes intransponíveis no mapa	X		
Ativar/desativar o modo aleatório para a sequência de peças/tubos	X		

Cliente			
Funcionalidades	Implementada	Implementada Parcialmente	Não Implementada
Clique com o botão esquerdo do rato em cima de uma célula onde ainda não passou a água permite mudar a peça aí existente	X		
Mouseover sobre a célula onde a água se encontra atualmente suspende temporariamente o seu fluxo			X
Jogar 2 clientes simultaneamente			X (1cliente)