Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia de Informação e Sistemas



Sistemas Operativos II

Relatório – 1ª Meta

Carlos Pinho ⎼ 21220528

Ricardo Reis ⎼ 21240456

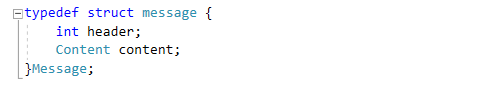
# Introdução

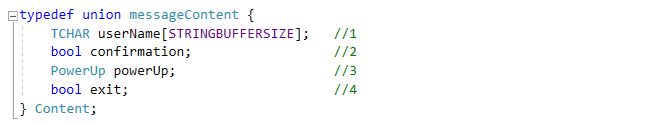
Este relatório tem como objetivo descrever de forma clara a implementação para esta meta.

Vai ser descrita a arquitetura utilizada para esta meta entre o cliente e o servidor e o porquê de a termos escolhido.

# Comunicação entre Servidor e Clientes Locais

Para o envio de mensagens entre ambos o Servidor e o Cliente utilizamos a estrutura *Message*. Dentro possui um header para identificar o tipo de mensagem e como deve ser lida e tem uma union, *Content,* com a mensagem. Decidimo-nos pela *union* pois nas mensagens só é envida um tipo de cada vez.



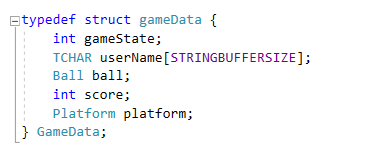


A comunicação é estabelecida através de dois buffers circulares do tipo Produtor-Consumidor em que um é utilizado para escrita do Servidor e leitura do Cliente e o outro para a escrita do Cliente e leitura do servidor. Decidimos fazer assim pois um único buffer em que ambos produzem e consomem poderia torna-se mais propenso a bugs. A contrapartida é que ficamos com o dobro dos handles de semaforos e mutexes. A inicialização dos 8 handles respetivos a esta arquitetura todos é feita pelo servidor na função *initialization()* podendo ser depois abertos pelo Cliente





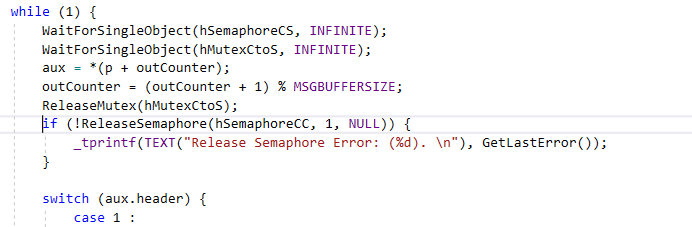
A zona de dados do jogo possui uma estrutura *GameData* que contém a informação do jogo que é escrita pelo Servidor e lida pelo Client

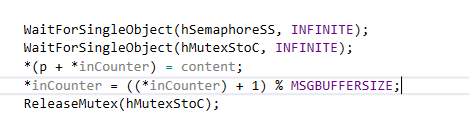


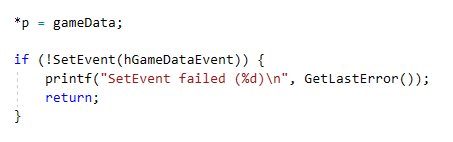
## Lado do Servidor

A recepção das mensagens do lado do servidor é feita por uma thread de espera passiva,



que consome dados do buffer quando o cliente os produz e depois os trata consoante o header da mensagem através de um *switch case.* 

Para enviar mensagens para o cliente utilizamos a função sendMessage(…) que espera que haje espaço vazio no buffer para enviar. Faz este controlo através dos semaforos e mutexes.

Para escrever na zona de dados de jogo utilizamos a thread UpdateGameData(…) que altera a estrutura do jogo na memório e notifica o cliente através de um Evento. 

## Lado Cliente

A arquitetura de mensagens do Cliente para o Servidor é exactamente a inversa da descrita em cima visto que temos um segundo buffer para esta. Por isso temos as mesmas funções de enviar e receber estando estas na DLL.

