Instituto Superior de Engenharia de Coimbra

Licenciatura em Eng. Informática

Relatório do Trabalho Prático de SO 20/21
Sistema CHAMPION

Meta 1

Diogo Miguel Pinto Pascoal, 2018019825

Nuno Gabriel Tavares Honório, 2019126457

1. Introdução

O trabalho prático de Sistemas Operativos consiste na implementação de um sistema de gestão de campeonatos de jogos denominado CHAMPION. Este sistema encarrega-se de fazer a ponte entre os jogadores e os jogos, mediando as mensagens trocadas entre ambos e gerindo o campeonato.

Este trabalho é concretizado em linguagem C para a plataforma Unix (Linux), sendo que nesta primeira meta apenas é explicado levemente as estruturas de dados planeadas e algumas ações por parte do árbitro.

2. Estruturas de Dados

Em vez de inserir todas as estruturas de dados num único ficheiro, foi decidido inserir cada estrutura num header correspondente, sendo que as estruturas de dados associadas ao árbitro encontram-se em árbitro.h, as que estão associadas ao cliente em cliente.h e assim seguindo. No caso de ser preciso uma estrutura de dados associada a outro elemento do sistema, não existe problema em incluir o header do mesmo.

2.1 Árbitro

O árbitro, de momento, conta apenas com duas estruturas de dados: VARS e infoArbitro:

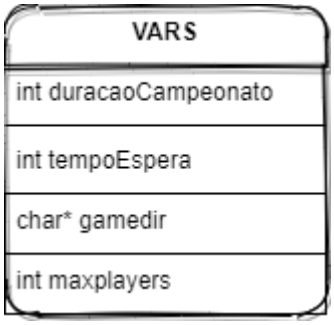

Figura 1 - Estrutura de Dados VARS

Figura 2 - Estrutura de Dados infoArbitro

VARS irá conter informação variável à execução do sistema: a duração do Campeonato e o Tempo de Espera, sendo que ambas as variáveis serão recebidas como argumentos aquando da execução do árbitro[1]; Também contém o "gamedir", sendo este o caminho dentro do sistema operativo onde se encontram os jogos para o sistema CHAMPION, fazendo este parte das variáveis que serão procuradas nas variáveis de ambiente[2]. Do mesmo modo, a variável maxplayers, que representa quantos jogadores o campeonato pode ter no máximo, é obtida através da variável de ambiente com o mesmo nome.[2]

2.2 Jogo

De momento o jogo é definido por uma estrutura GAME que contém apenas 3 inteiros: idJogo para saber de que jogo se trata, duracao e pontuacao, sendo que não nos é ainda claro na primeira meta o que esta parte do sistema necessita.

[1] Como sugerido no enunciado, é dado uso à função getopt() para a leitura dos argumentos
[2] Como sugerido no enunciado, a procura pelas variáveis de ambiente é feita através do getenv()

2.3 Cliente

O cliente (jogador), cuja estrutura de dados é PLAYER, é representando pelo seu nome e um ID que lhe é atribuído pelo sistema.

3. Valores por Omissão

Foram estabelecidos no header arbitro.h constantes que servem de padrão caso não seja possível obter o valor das variáveis presentes na estrutura VARS:

- duracaoCampeonato = 300000 (ms)
- tempoEspera = 60000 (ms)
- gamedir = ~/gamedir/
- maxplayers = 30

Cada uma destas variáveis só é usada se ela própria não puder ser encontrada. Se o programa só conseguir ler uma fração das variáveis (p.e. consegue encontrar duracaoCampeonato e tempoEspera mas não encontra gamedir nem maxplayers nas variáveis de ambiente), irá aceitar aquelas que foi possível ler e inserir os valores por omissão daquilo que não foi possível obter o valor.

4. MakeFile

O nosso MakeFile contém 5 targets: all, cliente, arbitro, jogo e clean. Ao escrever make cliente, make arbitro ou make jogo, será compilada apenas essa parte do sistema. Make all irá compilar os 3 programas e make clean irá limpar os executáveis mas manter os ficheiros de código.

```
all: cliente arbitro jogo

cliente: cliente.c
    gcc cliente.c  -o cliente.o
arbitro: arbitro.c
    gcc arbitro.c  -o arbitro.o
jogo: jogo.c
    gcc  jogo.c -o jogo.o

clean:
    rm -f *.o
```

Figura 3 – Ficheiro MakeFile

5. Setpath.sh

Também foi criado um pequeno script para dar export às duas variáveis de ambiente GAMEDIR e MAXPLAYERS cujo valores são os mesmo que em “Valores por Omissão”.