ESCOLA SUPERIOR DE TECNOLOGIA - IPS  
Ano Letivo 2015 / 2016  
  
Engenharia de Informática   
  
Sistemas Operativos. Profª Rossana Santos



Manual Técnico

Miguel Furtado 120221006 Turma G-02

Índice

Introdução..………………………………………………………………………………………….….2

Multibol………….………………………………………………………………………………………..3

Funcionamento da Simulação…………………………………………………………………..4

Métodos Mais Importante………………………………………………………………………..6

Semáforos………………………………………………………………………………………………..8

Mémoria Partilhada………………………………………………………………………………….9

Análise das Limitações……………………………………………………………………………10

Código Fontes…………………………………………………………………………………………11

Conclusão……………………………………………………………………………………………….17

Introdução

Este projeto foi desenvolvido para a unidade curricular de Sistemas Operativos da Licenciatura em Engenharia Informática Escola Superior de Tecnologia de Setúbal a pedido da docente Rossana Santos.

O projeto consiste no desenvolvimento de um jogo que foi chamado de Multibol. O Multibol é uma aplicação que pretende simular um jogo em que os jogadores irão marcar golos nas balizas dos adversários não podendo marcar golos na própria baliza.

Para o desenvolvimento do projeto foi usado a linguagem de programação C, em Linux, usando para a criação dos diferentes processos a função fork(), funções para a criação e manipulação de memória partilhada e para a sincronização destes processos foram usados semáforos .

Esta aplicação produz o output do jogo em real de simulação, para que deste modo o utilizador consiga visualizar o decorrer o jogo.

Multibol

O multibol é um jogo com um mínimo de três jogadores e uma bola podendo existir n Jogadores e m bolas, não podendo existir mais bolas do que jogadores. As bolas são distribuídas aleatoriamente pelos vários jogadores no início do jogo e cada jogador tem uma e só uma baliza.

Este irá decorrer ao longo de 30 segundos reais, o que equivale a 30 minutos de jogo. Após este tempo o jogo acaba e ganha quem tiver marcado mais golos nas balizas dos adversários. No fim irá ser mostrado o número de golos de cada jogador, o número de remates e a precisão dos remates relativamente ao número de golos.

Funcionamento da simulação

Inicialmente a simulação pede ao utilizador o número de jogadores que irá haver na mesma Fig. 1.



Fig. 1 Escolher jogadores

De seguida pede ao utilizador o número de bolas que irá haver na simulação Fig. 2.



Fig. 2 Escolher número de bolas

Após introduzir o número de jogadores e o número de bolas que irá haver na simulação e irá decorrer como se pode ver na Fig. 3.

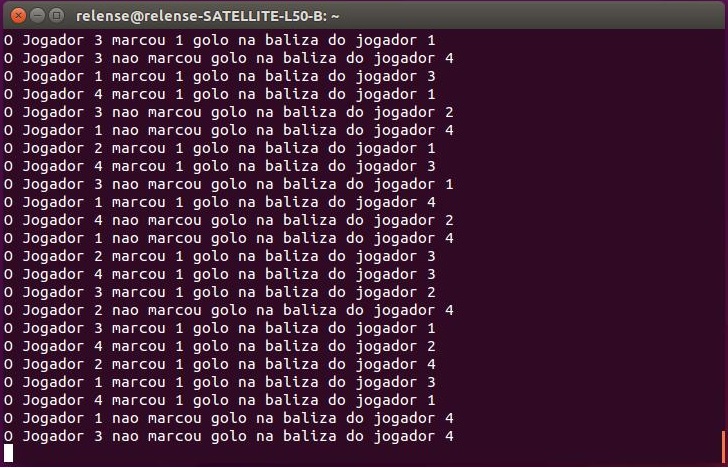


Fig. 3 Decorrer da simulação

Passado 30 segundos reais, ou seja, 30 minutos de jogo, este chega ao fim e mostra as estatísticas finais Fig. 4.

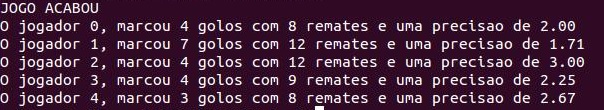


Fig. 4 Fim da simulação

Métodos Mais Importantes

* **void jogador(int i)**

Método que dá autorização para um jogador rematar se tiver bolas em sua posse, caso não tenha o jogador irá ficar à espera até ter autorização.

**Código Fonte:**

void jogador(int i)/\* metodo que chama cada jogador para jogar\*/

{

while(1){

P(semJogadores[i]);/\* se == 1 entra para rematar \*/

usleep((unsigned int) rand() % TEMPOESPERA);

rematar(i);

}

}

* **void rematar(int i)**

Método que remata a bola de um jogador em particular. Inicialmente este começa por criar um número aleatório entre 0 e o número de jogadores, que irá ser o jogador que vai receber o remate. Após isto é gerado um número aleatório entre 0 e 1, em que 1 é igual a ter marcado golo e zero é ter falhado. Quando o jogador marca golo, é registado que este tem mais 1 golo. Após isto guarda-se um remate para o jogador que rematou, remove-se uma bola ao jogador que rematou e acrescenta-se ao bola ao jogador que recebeu o remate.

**Código Fonte:**

void rematar(int i)/\* metodo para o jogador rematar a bola \*/

{

int receptor;

do{

receptor = rand() % players; /\* escolhe o jogador para quem vai rematar \*/

}while(receptor == i);

int golo = rand() % 2;/\* remata e tem 50% de hipotesse \*/

if(golo == 1){

printf("O Jogador %d marcou 1 golo na baliza do jogador %d\n", i, receptor);

guardarGolos(i);

}else{

printf("O Jogador %d nao marcou golo na baliza do jogador %d\n", i, receptor);

}

guardarRemates(i);

removerBola(i);

acrescentarBola(receptor);

}

Semáforos

Para o desenvolvimento desta simulação foram usados dois semáforos:

**mutex** – Semáforo binário que varia entre 0 e 1 com o objetivo de proteger o acesso a diferentes zonas críticas onde é usada memória partilhada.

**semJogadores[JOGADORES]** – Semáforo que varia entre 0 e o número de bolas que os diferentes jogadores têm em sua posse. Este semáforo é usado para restringir ou dar acesso aos diferentes jogadores de poderem rematar. Sendo que se for 0 não pode rematar e se for maior que 0 já pode.

Memória Partilhada

Nesta simulação foram usadas as seguintes variáveis de memória partilhada em que todas têm o tamanho do número de jogadores.

**int \*jogadores** – Usada para guardar as bolas dos respetivos jogadores.

**int \*golos** – Usada para guardar os golos de cada jogador.

**int \*remates** – Usada para guardar os remates de cada jogador

**float \*precisão** – Usada para guardar a precisão de remate de cada jogador.

Análise das limitações

Com o desenvolvimento desta aplicação deparei-me com algumas limitações e dificuldades sendo que algumas foram em lidar com floats e com o seu funcionamento para guardar valores, como parar um processo sem que este termine de modo a poder levar o jogo a prolongamento e penalties.

Como tal não foi possível desenvolver a aplicação para que esta ficasse a 100 por cento em todos os cenários, mas foi estendida de modo a conseguir implementar alguns dos requisitos pedidos.

Código Fonte

#include "sema.h"

#include <sys/shm.h>

#define JOGADORES 100

#define LIFETIME 30

#define SHMKEY 100

#define SHMKEY2 102

#define SHMKEY4 104

#define SHMKEY5 105

#define TEMPOESPERA 1000500

semaphore mutex, semJogadores[JOGADORES];

int \*golos;

int \*remates;

int \*jogadores;

float \*precisao;

int players, nBolas;

void nJogadores();

void numeroBolas();

void distribuirBolas();

void bolas();

void jogador(int i);

void rematar(int i);

void guardarGolos(int i);

void guardarRemates(int i);

void acrescentarBola(int i);

void removerBola(int i);

void calcularPrecisao();

void estatisticas();

main ()

{

nJogadores();

numeroBolas();

int shmid = shmget(SHMKEY, JOGADORES, 0777|IPC\_CREAT);/\* array de jogadores para guardar as bolas de cada jogador \*/

char \*addr = (char \*) shmat(shmid,0,0);

jogadores = (int\*) addr;

int shmid2 = shmget(SHMKEY2, JOGADORES, 0777|IPC\_CREAT);/\* array para guardar os remates de cada jogador \*/

char \*addr2 = (char \*) shmat(shmid2,0,0);

remates = (int\*) addr2;

int shmid4 = shmget(SHMKEY4, JOGADORES, 0777|IPC\_CREAT);/\* array para guardar os golos de cada jogador \*/

char \*addr4 = (char \*) shmat(shmid4,0,0);

golos = (int\*) addr4;

float shmid5 = shmget(SHMKEY5, JOGADORES, 0777|IPC\_CREAT);/\* array para guardar a precisao de remate de cada jogador \*/

char \*addr5 = (char \*) shmat(shmid5,0,0);

precisao = (float\*) addr5;

int child\_pid [players], /\* process ID's \*/

wait\_pid; /\* pid of terminated child \*/

int i, j, /\* loop variables \*/

child\_status; /\* return status of child \*/

for(i = 0; i < players; ++i){

semJogadores[i] = init\_sem(0); /\* iniciar o semafero de cada jogador 0 \*/

}

mutex = init\_sem(1);/\* iniciar semafero a 1 \*/

srand(time(NULL));

distribuirBolas();/\* atribuir bolas a jogadores \*/

bolas();/\* imprimir quem tem quantas bolas \*/

sleep(2);/\* espera 2 segundos antes de começar o jogo \*/

for (i = 0; i < players; ++i)

wait\_pid = wait (&child\_status); if (wait\_pid == -1)

{

perror ("wait failed");

} else { printf("Child %d terminated with code %d\n", wait\_pid, child\_status);

};

};

printf ("All child processes have terminated.\n");

printf("JOGO ACABOU\n");

calcularPrecisao();

estatisticas();

rel\_sem (mutex);

for(i = 0; i < players; i++){ rel\_sem(semJogadores[i]);

}

shmdt(addr);

shmctl(shmid, IPC\_RMID, NULL);

shmdt(addr2);

shmctl(shmid2, IPC\_RMID, NULL);

shmdt(addr4);

shmctl(shmid4, IPC\_RMID, NULL);

shmdt(addr5);

shmctl(shmid5, IPC\_RMID, NULL);

};

}; /\* end switch \*/

}; /\* end for \*/

}

void nJogadores()/\* pede o numero de jogadores ao utilizador \*/

{

do{

printf("Quantos jogadores quer?\n");

scanf("%d", &players);

}while( players > 100 || players < 3);

}

void numeroBolas()/\* pede o numero de bolas ao utilizador \*/

{

do{

printf("Quantas bolas ha em jogo?\n");

scanf("%d", &nBolas);

}while(nBolas > players);

}

void distribuirBolas()/\* metodo para dar bolas a cada jogador aleatorio \*/

{

int q;

for(q = 0; q < nBolas; ++q)

{

int jogadorAleatorio = rand() % players;/\* escolhe jogador aleatorio \*/ int valor = \*(jogadores + jogadorAleatorio); /\* guarda o numero de bolas no jogadorAleatorio \*/

\*(jogadores + jogadorAleatorio) = valor + 1; /\* acrescenta uma bola ao jogadorAleatorio \*/

V(semJogadores[jogadorAleatorio]);

}

}

void bolas()/\* imprimi as bolas que cada jogador tem \*/

{

int q;

for(q = 0; q < players; ++q){

if(\*(jogadores + q) > 0){

printf("O jogador %d tem %d bolas\n", q, \*(jogadores + q));

}

}

}

void jogador(int i)/\* metodo que chama cada jogador para jogar\*/

{

while(1){

P(semJogadores[i]);// se == 1 entra para

usleep((unsigned int) rand() % TEMPOESPERA);

rematar(i);

}

}

void rematar(int i)/\* metodo para o jogador rematar a bola \*/

{

int receptor;

do{

receptor = rand() % players; /\* escolhe o jogador para quem vai rematar \*/

}while(receptor == i);

int golo = rand() % 2;/\* remata e tem 50% de hipotesse \*/

if(golo == 1){

printf("O Jogador %d marcou 1 golo na baliza do jogador %d\n", i, receptor);

guardarGolos(i);

}else{

printf("O Jogador %d nao marcou golo na baliza do jogador %d\n", i, receptor);

}

guardarRemates(i);

removerBola(i);

acrescentarBola(receptor);

}

void guardarGolos(int i)/\* guarda os golos do jogador i \*/

{

P(mutex);

int valor = \*(golos + i);//verifica quantos golos tem o jogador que rematou

\*(golos + i) = valor + 1;/\* incrementa um golo para o jogador que rematou \*/

V(mutex);

}

void guardarRemates(int i)/\* guarda o remate que foi feito no respectivo jogador \*/

{

P(mutex);

int valor = \*(remates + i);/\* quantos remates já fez o que rematou \*/

\*(remates + i) = valor + 1;/\* incrementa um remate no jogador que rematou\*/

V(mutex);

}

void removerBola(int i)/\* remove uma bola \*/

{

P(mutex);

int valor = \*(jogadores + i);//vai buscar o numero de bola do jogador i

\*(jogadores + i) = valor - 1;//remove uma bola

V(mutex);

}

void acrescentarBola(int i)/\* acrescenta uma bola ao jogador i \*/

{

P(mutex);

int valor = \*(jogadores + i);/\* Quantas bolas tem o jogador que sofreu remate \*/

\*(jogadores + i) = valor + 1;/\* acrescenta um bola ao jogador que sofreu o remate \*/

V(mutex);

V(semJogadores[i]);

}

void estatisticas()/\* imprimi os resultados finais \*/

{

int k;

for(k = 0; k < players; ++k){

printf("O jogador %d, marcou %d golos com %d remates e uma precisao de %.2f\n", k, \*(golos + k), \*(remates + k), \*(precisao + k));

}

}

void calcularPrecisao()/\* calcula a precisao de remates de cada jogador \*/

{

int x;

for(x = 0; x < players; ++x){

P(mutex);

int valorRemates = \*(remates + x);

int valorGolos = \*(golos + x);

if(valorGolos > 0 && valorRemates > 0){

\*(precisao + x) = ((float) valorRemates / (float) valorGolos);

}else{

\*(precisao + x) = 0;

}

V(mutex);

}

}

Conclusão

O desenvolvimento deste projeto permitiu-nos conhecer mais sobre a linguagem de programação C, que anteriormente não havia adquirido conhecimentos. Como tal foi-me possível ficar a perceber mais sobre o funcionamento de processos e de como estes interagem uns com os outros num sistema operativo, seja individualmente ou em grupo e em como estes operam em conjunto com a memória partilhada de uma forma organizada.

A par do conhecimento sobre a linguagem C este projeto permitiu ter novos conhecimentos relativamente ao sistema operativo Linux, nomeadamente o Ubunto.