ANO LETIVO: 2015/2016

DOCENTE: ROSSANA SANTOS

Instituto Politécnico de Setúbal

Escola Superior de Tecnologias de Setúbal

MULTIBOL

MANUAL TÉCNICO

- SISTEMA OPERATIVOS -

Trabalho Realizado por:

André Bastos – 140221017

Luís Mestre – 140221002

Turma 2ºINF-ES-03/04

Índice

Introdução

Este manual tem como objetivo a descrição e explicação de todas as técnicas e métodos usados no projeto multibol no âmbito da disciplina de Sistemas Operativos.

A aplicação trata-se de um simulador de um jogo semelhante ao futebol, mas onde introduzimos o número de jogadores que queremos e pode haver mais do que uma bola em jogo ao mesmo tempo. Cada jogador tem também uma baliza associada.

A aplicação foi desenvolvida em linguagem C no sistema operativo Linux, utilizando técnicas como os semáforos e memória partilhada.

Objetivos

O desenvolvimento da aplicação Multibol teve em conta vários objetivos, entre eles são o jogo correr durante 30 minutos (30 segundos em tempo real), contabilizar o número de golos marcados feitos por cada jogador, para no final verificar quem foi o vencedor do jogo. Além deste objetivo, iremos verificar também os seguintes:

- ## Pódio

O pódio tem a listagem de todos os jogadores, que participaram no jogo, ordenados pelo número de golos marcados durante o jogo.

- ## Número de Remates

Pode-se verificar quantas vezes cada jogador rematou a bola ao longo do jogo. Sempre que a bola for rematada, esse valor será incrementado.

- ## Precisão de Remates

A precisão de remates dos jogadores será a razão entre o número de remates efetuados e os golos marcados. A probabilidade que cada jogador tem de marcar um golo é 50%.

- ## Faltas

Um jogador comete falta quando o tempo de remate ultrapassa o estipulado.

Código da Aplicação

Ficheiros e constantes

Temos os ficheiros de suporte necessários para a execução do programa.

```
#include "sema.h"
#include <sys/shm.h> ***
```

Definimos 3 constantes: máximo número de jogadores, máximo número de bolas e também do número de processos.

Há um limite máximo de 100 para cada um destes, se despejarmos aumentar o número de cada uma destas componentes, teríamos que alterar o valor destes atributos.

```
#define MAX_PLAYERS     100
#define MAX_BOLAS       100
#define MAX_CHILD       100
```

Está definido também uma constante LIFETIME que representa o tempo que tem cada jogo (30 segundos), tendo em conta que cada segundo é considerado um minuto para o projeto.

```
#define LIFETIME        30
```

Temos as keys de acesso a cada memória partilhada, sendo a SHMKEY para a memória que guarda o número de bolas que tem cada jogador, a **SHMKEY1** para a memória que guarda os golos de cada jogador que já marcou, a **SHMKEY2** para a memória que guarda o número de remates de cada jogador e a **SHMKEY3** para a memória que guarda as faltas cometidas por cada jogador.

```
#define SHMKEY (key_t) 0x10
#define SHMKEY1 (key_t) 0x11
#define SHMKEY2 (key_t) 0x12
#define SHMKEY3 (key_t) 0x13
```

Atributos

O atributo **nrJogadores**, do tipo inteiro, guarda o valor do número de jogadores que o utilizador introduz na execução da aplicação.

O atributo **nrBolas**, do tipo inteiro, guarda o valor do número de bolas que o utilizador introduz na execução da aplicação.

O atributo **nrProcessos**, do tipo inteiro, guarda o valor do número de processos que iram ser criados. Este é composto por o número de jogadores que o utilizador introduz, juntamente com outros processos que sejam relevantes.

```
int nrJogadores;
int nrBolas;
int nrProcessos;
```

O semáforo **mutex** é o semáforo que irá restringir o acesso às memórias partilhadas para que os processos não acedam ao mesmo tempo e assim haver uma boa sincronização dos mesmos.

O array de semáforos **arraySema** tem o tamanho dado pela constante MAX_PLAYERS que guarda o número máximo de jogadores. Este array irá conter todos os semáforos que estão associados a cada jogador, seguindo a lógica de que cada índice é um jogador. Tendo também em conta que cada jogador agora tem um semáforo associado que vai informar se o jogador pode ou não rematar.

```
semaphore mutex;
semaphore arraySema[MAX_PLAYERS];
```

Temos ao todo 3 memórias partilhadas, todos eles com o mesmo tipo de estratégia mas com finalidades diferentes. Em todos eles cada índice representa cada jogador.

No ***jogadores** cada jogador (índice) guarda o número de bolas que possui naquele momento.

No ***golos** cada jogador (índice) guarda o número de golos já marcados.

No ***remates** cada jogador (índice) guarda o número de remates já feitos.

No ***faltas** cada jogador (índice) guarda o número de faltas cometidas.

```
int *jogadores;
int *golos;
int *remates;
int *faltas;
```

Métodos

O método **encontraAdversario** recebe o número de jogador para o qual queremos encontrar um jogador adversário.

O objetivo deste método é encontrar um adversário que não tenha o mesmo número de jogador que o atual, assim gerando um número aleatório que não seja maior que o número do último jogador e repetindo até que o número gerado não seja o do jogador atual.

```
int encontraAdversario(int jogadorAtual){
        int adv;
        do{
                adv  =rand() % nrJogadores;
        }while(adv == jogadorAtual);

    return adv;
}
```

O método **rematar** recebe o número de jogador atual, e o número de jogador de um adversário.

Imprime no ecrã uma frase que informa qual o jogador que recebeu a bola e o jogador que vai tentar defender. Gera um número aleatório que apenas pode tomar os valores de 0 ou 1.

Vai aceder à memória partilhada *jogadores e retira uma bola do jogador atual, e o número de remates (*remates) e incrementa um remate ao jogador atual,

O aleatório gerado representa a probabilidade de um jogador marcar ser de 50%.

Caso a variável tome o valor de 0, imprime no ecrã que o jogador marcou e adiciona à memória partilhada associada aos golos marcados (*golos) o golo que marcou tendo em conta o índice através do número de jogador.

Caso seja 1, imprime no ecrã que o jogador defendeu.

```
void rematar(int nrJogador, int jogadorAdv){

    printf("O jogador %d recebeu uma bola e vai rematar contra o jogador %d\n",nrJogador+1,jogadorAdv+1) ;
    int r = rand() % 2;

    *(jogadores+nrJogador) = *(jogadores+nrJogador)-1;
    *(jogadores+jogadorAdv) = *(jogadores+jogadorAdv)+1;
        *(remates+nrJogador) = *(remates+nrJogador)+1;

    if(r==0){
                printf("Marcou.\n");
                *(golos+nrJogador) = *(golos+nrJogador)+1;
    }
    if(r==1) printf("Defendeu.\n");

}
```

O método **jogador** recebe um inteiro que representa o número do jogador que vai jogar.

Através da operação P(arraySema[i]) decrementamos o semáforo associado ao jogador que vai jogar e o programa decide se a aplicação pode jogar.

De seguida gera-se um inteiro rand entre 0 e 600 que determina o tempo que o jogador demora a rematar, caso este tempo seja superior a 500, o jogador comete falta.

A operação usleep(tempo*1000) gera a quantidade de tempo que o jogador espera até rematar.

Guardamos numa variável (jogadorAdv), do tipo inteiro, o número do jogador adversário através do método encontraAdversario(i).

Restringimos o acesso às memórias partilhadas através da operação P(mutex).

Executa o método rematar e de seguida através da operação V(mutex) paramos de restringir o acesso às memórias partilhadas.

A operação V(arraySema[jogadorAdv]) vai incrementar o semáforo relativo ao jogador adversário, assim dizendo que esse jogador já pode rematar também.

```
void jogador(int i){

    P(arraySema[i]);
    int tempo = rand() % 601;
                    if (tempo > 500) {

                    *(faltas + i) = *(faltas + i) + 1;
                    printf("Jogador %d cometeu uma falta \n",i+1);
                    tempo = 500;

                    }


                    usleep(tempo * 1000);

    int jogadorAdv = encontraAdversario(i);
    P(mutex);
    rematar(i, jogadorAdv);
    V(mutex);
    V(arraySema[jogadorAdv]);

}
```

O método **atribuirBolas** é suposto ser chamado apenas no início do programa e tem o objetivo de atribuir as bolas todas aos jogadores aleatoriamente.

Para tal recorremos ao uso de um ciclo que continua a repetir o código enquanto a variável count for diferente do número de bolas.

Guardamos um número aleatório na variável jogador, tendo um máximo do número de jogadores existentes. Acedemos à memória partilhada *(jogadores) e incrementamos o número de bolas que esse jogador tem. Não é preciso o uso de um semáforo para restringir o acesso às memórias partilhadas pois este método destina-se apenas a ser chamado no início do programa, onde os processos ainda nem estão criados.

Verificamos se esse jogador já tem uma bola, caso ele já tenha uma bola além da bola que lhe demos recentemente então retiramos a bola que lhe foi dada na memória partilhada. Pois no início do jogo cada jogador apenas pode ter 0 ou 1 bolas.

Se for a primeira bola, incrementamos o count e também o semáforo associado para indicar que este jogador pode rematar.

```
void atribuirBolas(){

        int count =0;
        int jogador=0;

        do{

                jogador = rand() % nrJogadores;

                *(jogadores+ jogador)= *(jogadores+ jogador)+1;

                if(*(jogadores+ jogador) >= 2){
                        *(jogadores+ jogador)= *(jogadores+ jogador)-1;
                }else{
                        count= count+1;

                        V(arraySema[jogador]);
                }
        }while(count != nrBolas);
}
```

O método **iniciarSemaforos** tem o objetivo de inicializar todos os semáforos associados com cada jogador, no arraySema.

```
void iniciarSemaforos(){
    int i;
        for(i =0 ; i<nrJogadores; i++){
                arraySema[i] = init_sem(0);
        }
}
```

O método **shellSort** contem um algoritmo com o destino a ordenar os jogadores que tiveram mais golos para o pódio.

```
void shellSort(int *vet, int size, int *jogador) {
    int i , j , value, value2;
    int gap = 1;
    while(gap < size) {
        gap = 3*gap+1;
    }
    while ( gap > 1) {
        gap /= 3;
        for(i = gap; i < size; i++) {
            value = vet[i];
                        value2 = jogador[i];
                        j = i - gap;
            while (j >= 0 && value < vet[j]) {
                vet [j + gap] = vet[j];
                                jogador[j + gap] = jogador[j];
                j -= gap;
            }
            vet [j + gap] = value;
                        jogador[j+gap]= value2;
        }
    }
}
```

O método **isProlongamento** faz a verificação no final do jogo (passados os 30 segundos) se iria ou não haver prolongamento. Verifica se existe mais do que 1 jogador com o maior número de golos, caso haja mais que 1 retorna o valor 1, caso não haja retorna 0.

```
int isProlongamento(){
    int arrayGolos[nrJogadores];
    int arrayNumJogadores[nrJogadores];
    int i = 0;
    for( ; i<nrJogadores; i++){

        arrayNumJogadores[i]= i+1;
        arrayGolos[i]=*(golos+i);
    }

    shellSort(arrayGolos,nrJogadores,arrayNumJogadores);
    i = nrJogadores-1;
    int maiorNumGolos=arrayGolos[i];
    int count = 0;
    for(;i>-1;i--){
        if(maiorNumGolos == arrayGolos[i]){
            count++;
        }
    }

    if(count>1){
        return 1;
    }else{
        return 0;
    }
}
```

O método **podio** destina-se a imprimir no ecrã o lugar em que cada jogador ficou e também os golos que cada um marcou, com o suporte de novos arrays e também do método shellsort.

```
void podio(){
        int array[nrJogadores];
        int arrayNumJogadores[nrJogadores];
        int i=0;
        for( ; i<nrJogadores; i++){

                arrayNumJogadores[i]= i+1;
                array[i]=*(golos+i);
        }

        shellSort(array,nrJogadores,arrayNumJogadores);
        i=nrJogadores-1;
        for(;i>-1;i--){
                printf("O jogador que está em %d lugar é o jogador %d, com %d golos marcados.\n",
 nrJogadores - i, arrayNumJogadores[i], array[i]);
        }
}
```

O método **nrDeRemates** imprime no ecrã o número de remates feitos por cada jogador.

```
void nrDeRemates(){
        int i =0;
        for(;i<nrJogadores;i++){
                printf("O jogador %d rematou %d vezes.\n", i+1, *(remates + i));
        }
}
```

O método **precisaoRemates** imprime no ecrã a percentagem de precisão de cada jogador, utilizando assim uma fórmula.

```
void precisaoRemates(){
        int i =0;
        for(;i<nrJogadores;i++){
                if(*(remates+i) == 0){
                printf("O jogador %d não chegou a rematar.\n", i+1 );
                }else{
                float precisao = (float)  *(golos + i)/ *(remates + i);
                int percentagem = (int)(precisao * 100);
                printf("A precisão do jogador %d foi de %d%c \n", i+1,percentagem , '%');
                }
        }
}
```

O método **nrFaltas** imprime no ecrã o número de faltas cometidas por cada jogador.

```
void nrFaltas(){
    int i =0;
    for(;i<nrJogadores;i++){
        printf("O jogador %d fez %d faltas.\n", i+1, *(faltas + i));
    }
}
```

O método **bolasAtribuidas** verifica se as bolas já foram distribuídas ou não pelos jogadores. Caso tenha sido atribuídas retorna 1, caso contrário retorna 0.

```
int bolasAtribuidas(){
    int count =0;
    int i=0;
    for(;i<nrJogadores;i++){
        if(*(jogadores + i) == 1){
            count++;
        }
    }
    if(count==nrBolas){
        return 1;
    }else{
        return 0;
    }
}
```

Main

O **main** começa por pedir ao utilizador o número de jogadores e de bolas que deseja e guarda nas respetivas variáveis. O atributo **nrProcessos** terá o valor do número de jogador mais algum processo relevante. Se tencionávamos acrescentar um árbitro por exemplo, então teriamos que acrescentar em 1 esta variável.

```
printf("Insira o número de jogadores (não deverá ser maior que 100): ");
scanf("%d", &nrJogadores);

nrProcessos = nrJogadores;

printf("Insira o número de bolas (não deverá ser maior que %d): ",nrJogadores);
scanf("%d", &nrBolas);
```

As linhas de código abaixo têm o objetivo de alocar o espaço para cada memória partilhada.

```
int shmid = shmget(SHMKEY, nrJogadores, 0777|IPC_CREAT);
char *addr = (char *) shmat(shmid,0,0);
jogadores = (int*) addr;

int shmid1 = shmget(SHMKEY1, nrJogadores, 0777|IPC_CREAT);
char *addr1 = (char *) shmat(shmid1,0,0);
golos = (int*) addr1;

int shmid2 = shmget(SHMKEY2, nrJogadores, 0777|IPC_CREAT);
char *addr2 = (char *) shmat(shmid2,0,0);
remates = (int*) addr2;

int shmid3 = shmget(SHMKEY3, nrJogadores, 0777|IPC_CREAT);
char *addr3 = (char *) shmat(shmid3,0,0);
faltas = (int*) addr3;
```

Inicializamos variáveis necessárias à criação de processos e também iniciamos o semáforo **mutex**.

```
int             child_pid [MAX_CHILD],
        wait_pid;
int             i, j,
        child_status;

mutex = init_sem(1);
```

Chamamos os métodos que vão iniciar os semáforos dos jogadores e atribuir as bolas aos jogadores. Nestes métodos acedemos à memória partilhada mas não é preciso restringir o acesso pois ainda nem foram criados processos.

```
iniciarSemaforos();
```

No código a seguir será onde irá necessariamente criar os processos.

Por cada processo, até ao número de jogadores, será executado o código referente aos jogadores.

O método **atribuirBolas** é chamado no processo pai porque assim garante uma distribuição aleatória das bolas e não será corrido pelos processos filhos.

```
for (i = 0; i < nrProcessos; ++i)
{
    child_pid [i] = fork ();

    switch (child_pid [i])
    {
        case -1:
            perror ("fork failed");
            exit (1);
            break;
        case 0:
            alarm(LIFETIME);

            for(;;)
            {

            if(i<nrJogadores) jogador(i);

            }
            break;
        default:
            if(bolasAtribuidas() == 0){
                srand((int) time(NULL));
                atribuirBolas();
            }

            if (i == (nrProcessos - 1))
            {
                for (j = 0; j < nrProcessos; ++j)
                {
                    wait_pid = wait (&child_status);
                    if (wait_pid == -1)
                    {
                        perror ("wait failed");
                    };
                };
```

Ainda no processo “pai”, ou seja, no final do jogo chamamos cada um dos métodos que imprime no ecrã as estatísticas.

```
puts("\nO Jogo acabou.\n");

if(isProlongamento() == 1) puts("Haveria prolongamento\n");

puts("------PODIO------");
podio();

puts("\n------NUMERO DE REMATES------");
nrDeRemates();

puts("\n------PERCENTAGEM DE PRECISAO DOS REMATES------");
precisaoRemates();

puts("\n------NUMERO DE FALTAS------");
nrFaltas();
```

Mais abaixo temos a libertação de todos os semáforos do programa, nomeadamente o **mutex** e todos os que estão no **arraySema**.

```
rel_sem (mutex);


int i=0;
for( ; i<nrJogadores; i++){
rel_sem(arraySema[i]);
}
```

Depois ainda temos a libertação do espaço destinado às memórias partilhadas.

```
shmdt(addr);
shmdt(addr1);
shmdt(addr2);
shmdt(addr3);
shmctl(shmid, IPC_RMID, NULL);
shmctl(shmid1, IPC_RMID, NULL);
shmctl(shmid2, IPC_RMID, NULL);
shmctl(shmid3, IPC_RMID, NULL);
```

Por último, temos o código que pergunta ao utilizador se quer recomeçar o jogo, e recomeça se for o caso.

```
printf("\nDeseja recomeçar o jogo?\n1-Sim\n2-Não\nR: ");
int resposta =0;
scanf("%d", &resposta);
if(resposta==1) main();
```

O **main** acaba aqui com o fecho de ciclos, bem como com a aplicação.

Análise de Limitações do programa (o que não conseguimos fazer)

Para além dos objetivos que descrevemos no início, houve outros que não referimos visto que não os conseguimos cumprir a tempo. Entre eles são:

- ## Prolongamento (incompleto) e Grandes Penalidades

Quando passado os 30 segundos (30 minutos) deveríamos de verificar se houve mais do que um jogador com o maior número de golos iguais. Caso isso acontecesse, os respetivos jogadores com a maior pontuação iriam disputar mais 5 segundos (5 minutos na aplicação).

Depois no final do prolongamento se ainda houver jogadores com o mesmo número de golos esses iriam disputar as grandes penalidades ("Morte Súbita") onde o primeiro a marcar é o vencedor.

No nosso jogo o que conseguimos fazer a tempo foi a verificação de que existe ou não prolongamento, mas os jogadores não chegam a ir a prolongamento.

- ## Faltas (incompleto)

Supostamente, uma falta é cometida quando o jogador, caso tenha mais do que 1 bola em posse, remata outra em vez da que tem há mais tempo em posse. No nosso jogo consideramos falta caso o jogador demore a rematar, independentemente da bola que escolher rematar.

Outro cenário que pode acontecer é por vezes no primeiro remate de cada jogador com bola se imprimir primeiro as faltas que cada jogador fez antes de imprimir o remate

- ## Resultados a obter

Tendo em conta os 2 pontos que foram referidos anteriormente, no final de um jogo não iriamos puder dispor o tempo de posse de bola de cada jogador, dizer se foi ou não necessário a prolongamento e a grandes penalidades e também não conseguimos dizer quantas faltas cada jogador cometeu.

- ## Remates

Reparamos que um dos "erros" que temos no jogo é que nos primeiros remates dos jogadores que têm inicialmente bola, eles acabam por rematar para um jogador em específico (excepto esse mesmo jogador que remata para outro).

- ## Posse de Bola

Apesar de pudermos fazer um contagem do tempo que o jogador demora a rematar, não conseguimos fazer a contagem do tempo desde que ele recebe uma bola até ao momento do remate.

Resultados e Estatísticas

NOTA: resultados obtidos num jogo com 50 jogadores e 50 bolas.

- Início do jogo

```
Insira o número de jogadores (não deverá ser maior que 100): 50
Insira o número de bolas (não deverá ser maior que 50): 50
O jogador 1 recebeu uma bola e vai rematar contra o jogador 37
Marcou.
O jogador 2 recebeu uma bola e vai rematar contra o jogador 10
Marcou.
O jogador 3 recebeu uma bola e vai rematar contra o jogador 10
Marcou.
O jogador 4 recebeu uma bola e vai rematar contra o jogador 10
Marcou.
O jogador 5 recebeu uma bola e vai rematar contra o jogador 10
Marcou.
O jogador 6 recebeu uma bola e vai rematar contra o jogador 10
Marcou.
O jogador 7 recebeu uma bola e vai rematar contra o jogador 10
Marcou.
O jogador 9 recebeu uma bola e vai rematar contra o jogador 10
Marcou.
O jogador 10 recebeu uma bola e vai rematar contra o jogador 13
Defendeu.
```

- Pódio

```
------PODIO------
O jogador que está em 1 lugar é o jogador 10, com 55 golos marcados.
O jogador que está em 2 lugar é o jogador 50, com 43 golos marcados.
O jogador que está em 3 lugar é o jogador 18, com 39 golos marcados.
O jogador que está em 4 lugar é o jogador 27, com 39 golos marcados.
O jogador que está em 5 lugar é o jogador 28, com 37 golos marcados.
O jogador que está em 6 lugar é o jogador 20, com 32 golos marcados.
O jogador que está em 7 lugar é o jogador 39, com 31 golos marcados.
O jogador que está em 8 lugar é o jogador 48, com 30 golos marcados.
O jogador que está em 9 lugar é o jogador 23, com 30 golos marcados.
O jogador que está em 10 lugar é o jogador 1, com 30 golos marcados.
O jogador que está em 11 lugar é o jogador 8, com 28 golos marcados.
O jogador que está em 12 lugar é o jogador 29, com 28 golos marcados.
O jogador que está em 13 lugar é o jogador 30, com 27 golos marcados.
O jogador que está em 14 lugar é o jogador 17, com 27 golos marcados.
O jogador que está em 15 lugar é o jogador 14, com 25 golos marcados.
O jogador que está em 16 lugar é o jogador 24, com 23 golos marcados.
O jogador que está em 17 lugar é o jogador 33, com 22 golos marcados.
O jogador que está em 18 lugar é o jogador 37, com 21 golos marcados.
O jogador que está em 19 lugar é o jogador 6, com 21 golos marcados.
O jogador que está em 20 lugar é o jogador 25, com 20 golos marcados.
O jogador que está em 21 lugar é o jogador 32, com 20 golos marcados.
O jogador que está em 22 lugar é o jogador 22, com 20 golos marcados.
O jogador que está em 23 lugar é o jogador 42, com 19 golos marcados.
O jogador que está em 24 lugar é o jogador 36, com 17 golos marcados.
O jogador que está em 25 lugar é o jogador 16, com 17 golos marcados.
O jogador que está em 26 lugar é o jogador 35, com 15 golos marcados.
O jogador que está em 27 lugar é o jogador 49, com 14 golos marcados.
O jogador que está em 28 lugar é o jogador 21, com 13 golos marcados.
O jogador que está em 29 lugar é o jogador 12, com 12 golos marcados.
O jogador que está em 30 lugar é o jogador 15, com 11 golos marcados.
O jogador que está em 31 lugar é o jogador 13, com 11 golos marcados.
O jogador que está em 32 lugar é o jogador 5, com 11 golos marcados.
O jogador que está em 33 lugar é o jogador 40, com 11 golos marcados.
O jogador que está em 34 lugar é o jogador 44, com 10 golos marcados.
O jogador que está em 35 lugar é o jogador 46, com 10 golos marcados.
O jogador que está em 36 lugar é o jogador 7, com 9 golos marcados.
O jogador que está em 37 lugar é o jogador 9, com 9 golos marcados.
O jogador que está em 38 lugar é o jogador 38, com 9 golos marcados.
O jogador que está em 39 lugar é o jogador 34, com 9 golos marcados.
O jogador que está em 40 lugar é o jogador 43, com 7 golos marcados.
O jogador que está em 41 lugar é o jogador 11, com 7 golos marcados.
O jogador que está em 42 lugar é o jogador 2, com 7 golos marcados.
O jogador que está em 43 lugar é o jogador 41, com 7 golos marcados.
```

- Número de Remates

```
------NUMERO DE REMATES------
O jogador 1 rematou 58 vezes.
O jogador 2 rematou 10 vezes.
O jogador 3 rematou 5 vezes.
O jogador 4 rematou 8 vezes.
O jogador 5 rematou 20 vezes.
O jogador 6 rematou 37 vezes.
O jogador 7 rematou 16 vezes.
O jogador 8 rematou 65 vezes.
O jogador 9 rematou 17 vezes.
O jogador 10 rematou 105 vezes.
O jogador 11 rematou 8 vezes.
O jogador 12 rematou 22 vezes.
O jogador 13 rematou 20 vezes.
O jogador 14 rematou 40 vezes.
O jogador 15 rematou 19 vezes.
O jogador 16 rematou 33 vezes.
O jogador 17 rematou 42 vezes.
O jogador 18 rematou 63 vezes.
O jogador 19 rematou 3 vezes.
O jogador 20 rematou 52 vezes.
O jogador 21 rematou 25 vezes.
O jogador 22 rematou 35 vezes.
O jogador 23 rematou 61 vezes.
O jogador 24 rematou 48 vezes.
O jogador 25 rematou 40 vezes.
O jogador 26 rematou 7 vezes.
O jogador 27 rematou 74 vezes.
O jogador 28 rematou 70 vezes.
O jogador 29 rematou 53 vezes.
O jogador 30 rematou 51 vezes.
O jogador 31 rematou 6 vezes.
O jogador 32 rematou 36 vezes.
O jogador 33 rematou 36 vezes.
O jogador 34 rematou 17 vezes.
O jogador 35 rematou 29 vezes.
O jogador 36 rematou 31 vezes.
O jogador 37 rematou 39 vezes.
O jogador 38 rematou 16 vezes.
O jogador 39 rematou 53 vezes.
O jogador 40 rematou 20 vezes.
O jogador 41 rematou 8 vezes.
O jogador 42 rematou 32 vezes.
O jogador 43 rematou 8 vezes.
O jogador 44 rematou 18 vezes.
O jogador 45 rematou 6 vezes.
O jogador 46 rematou 18 vezes.
```

- Percentagem de precisão dos remates

```
------PERCENTAGEM DE PRECISAO DOS REMATES------
A precisão do jogador 1 foi de 51%
A precisão do jogador 2 foi de 70%
A precisão do jogador 3 foi de 80%
A precisão do jogador 4 foi de 87%
A precisão do jogador 5 foi de 55%
A precisão do jogador 6 foi de 56%
A precisão do jogador 7 foi de 56%
A precisão do jogador 8 foi de 43%
A precisão do jogador 9 foi de 52%
A precisão do jogador 10 foi de 52%
A precisão do jogador 11 foi de 87%
A precisão do jogador 12 foi de 54%
A precisão do jogador 13 foi de 55%
A precisão do jogador 14 foi de 62%
A precisão do jogador 15 foi de 57%
A precisão do jogador 16 foi de 51%
A precisão do jogador 17 foi de 64%
A precisão do jogador 18 foi de 61%
A precisão do jogador 19 foi de 66%
A precisão do jogador 20 foi de 61%
A precisão do jogador 21 foi de 52%
A precisão do jogador 22 foi de 57%
A precisão do jogador 23 foi de 49%
A precisão do jogador 24 foi de 47%
A precisão do jogador 25 foi de 50%
A precisão do jogador 26 foi de 85%
A precisão do jogador 27 foi de 52%
A precisão do jogador 28 foi de 52%
A precisão do jogador 29 foi de 52%
A precisão do jogador 30 foi de 52%
A precisão do jogador 31 foi de 83%
A precisão do jogador 32 foi de 55%
A precisão do jogador 33 foi de 61%
A precisão do jogador 34 foi de 52%
A precisão do jogador 35 foi de 51%
A precisão do jogador 36 foi de 54%
A precisão do jogador 37 foi de 53%
A precisão do jogador 38 foi de 56%
A precisão do jogador 39 foi de 58%
A precisão do jogador 40 foi de 55%
A precisão do jogador 41 foi de 87%
A precisão do jogador 42 foi de 59%
A precisão do jogador 43 foi de 87%
A precisão do jogador 44 foi de 55%
A precisão do jogador 45 foi de 83%
A precisão do jogador 46 foi de 55%
A precisão do jogador 47 foi de 87%
```

- Número de Faltas

```
------NUMERO DE FALTAS------
O jogador 1 fez 7 faltas.
O jogador 2 fez 2 faltas.
O jogador 3 fez 1 faltas.
O jogador 4 fez 2 faltas.
O jogador 5 fez 4 faltas.
O jogador 6 fez 8 faltas.
O jogador 7 fez 4 faltas.
O jogador 8 fez 13 faltas.
O jogador 9 fez 4 faltas.
O jogador 10 fez 11 faltas.
O jogador 11 fez 2 faltas.
O jogador 12 fez 5 faltas.
O jogador 13 fez 4 faltas.
O jogador 14 fez 8 faltas.
O jogador 15 fez 4 faltas.
O jogador 16 fez 5 faltas.
O jogador 17 fez 8 faltas.
O jogador 18 fez 16 faltas.
O jogador 19 fez 1 faltas.
O jogador 20 fez 12 faltas.
O jogador 21 fez 5 faltas.
O jogador 22 fez 7 faltas.
O jogador 23 fez 13 faltas.
O jogador 24 fez 7 faltas.
O jogador 25 fez 7 faltas.
O jogador 26 fez 2 faltas.
O jogador 27 fez 13 faltas.
O jogador 28 fez 8 faltas.
O jogador 29 fez 12 faltas.
O jogador 30 fez 9 faltas.
O jogador 31 fez 1 faltas.
O jogador 32 fez 8 faltas.
O jogador 33 fez 8 faltas.
O jogador 34 fez 4 faltas.
O jogador 35 fez 6 faltas.
O jogador 36 fez 5 faltas.
O jogador 37 fez 6 faltas.
O jogador 38 fez 4 faltas.
O jogador 39 fez 10 faltas.
O jogador 40 fez 4 faltas.
O jogador 41 fez 2 faltas.
O jogador 42 fez 6 faltas.
O jogador 43 fez 2 faltas.
O jogador 44 fez 4 faltas.
O jogador 45 fez 2 faltas.
O jogador 46 fez 4 faltas.
O jogador 47 fez 2 faltas.
```