Relatório - Trabalho
Time de Basquete

Vitor Fernandes Dullens, 16/0148260

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CiC 117935 - Programação Concorrente

{vitordullens2.0}@gmail.com

1. Introdução

Foi passado um projeto que tem como objetivo o desenvolvimento de algoritmos para tratar problemas de comunicação entre processos através de memória compartilhada. Ou seja, utilizando os conhecimentos adquiridos na aula, deve-se produzir um problema capaz de ser solucionado com programação concorrente.

Este paradigma de programação consiste em programas capazes de executar de forma simultânea várias tarefas computacionais, elas podem ser implementadas como processos separados ou como um conjunto de threads dentro de um processo. Programação concorrente está focada na interação entre as tarefas. A interação e a comunicação correta entre as diferentes tarefas, além da coordenação do acesso concorrente aos recursos computacionais são as principais tarefas a serem desenvolvidas num sistema concorrente.

A partir dessas informações foi proposto a criação de um problema concorrente e o desenvolvimento de uma solução para o mesmo. Tal fato foi realizado, criando o problema do Time de Basquete, que é basicamente um jogo de basquete com jogadores comuns, jogadores muito bons(estrelas) e um juiz, o jogo deve obedecer algumas regras básicas como ter um número certo de jogadores em quadra para ser jogado e prioridades para definir quem entra ou não na partida.

Na seção de Formalização do Problema, este é abordado de forma mais aprofundada, na seção Solução do Problema é explicado sobre como ele foi solucionado dentro do paradigma proposto e na Conclusão é são feitos comentários finais e citadas dificuldades durante o processo.

2. Formalização do Problema

O problema do Time de Basquete se consiste em uma pseudo simulação de um jogo de basquete infinito, porém com alguns detalhes adicionais:

- Devem ter 5 jogadores em quadra para o jogo começar;
- Cada time possui pelo menos 1 estrela;

Uma estrela significa um jogador muito bom, decisivo e habilidoso, que não pode ficar muito tempo no banco, pois o time sempre precisa dele para pontuar melhor.

Tendo esses dados em vista, foi desenvolvido um problema baseado nisso:

- Precisamos de um juiz para verificar se exatamente 5 jogadores estao em quadra;
- Quando uma estrela está no banco ela possui prioridade para voltar para a quadra;

A partir disso, o problema foi criado, um Time de Basquete que possui pelo menos 4 jogadores comuns e 1 estrela, totalizando no mínimo 5 jogadores e 1 juiz capaz de iniciar o jogo e identificar possíveis substituições (quando um jogador sai e o outro entra), ou seja, 6 thread ao todo no mínimo. Com isso o problema começa com 5 jogadores (com a(s) estrela(s) inclusa(s)) entrando em quadra e os demais dos jogadores ficando no banco, após o apito inicial do Juiz, o jogo se inicia e cada jogador joga um tempo até cansar, depois desse tempo o jogador sai da partida e fica no banco descansando enquanto outro entra em seu lugar (substituição), neste caso, se uma estrela estiver no banco ela deve ter prioridade sobre os demais para entrar no jogo. Depois que cada jogador descansa ele fica disponível no banco para voltar ao jogo (ser substituído) quando algum outro jogador cansar.

3. Solução do Problema

No algoritmo foi utilizado a biblioteca pthread da linguagem C, para que fosse possível a criação de uma solução concorrente para o problema. Foram criadas 11 thread no total, 2 jogadores estrelas, 8 jogadores comuns e 1 juíz.

No programa existem três processos principais:

- Juiz;
- Estrela;
- Jogador;

Cada um deles possui sua tarefa dentro do jogo. O **Juiz** é resposável por autorizar o início da partida e contar as substituições, a **Estrela** joga, descansa e fica no banco e o **Jogador** joga, descnsa e fica no banco porém respeitando a prioridade da estrela para entrar em quadra.

Tendo isso em vista, o algoritmo foi criado, dentro da thread do jogador, antes de entrar em quadra, deve-se verificar se existe espaço na quadra, ou seja, não possui 5 jogadores e se nao tem nenhum estrela esperando para entrar. Se essas condições não forem estabelecidas, entao o jogador vai para o banco e aguarda até um espaço ser liberado, ou seja, um jogador sair da partida, como é posível ver no trecho de codigo abaixo:

```
while(em_quadra == 5 || estrela_espera > 0){
    no_banco++;

    print_dados();
    printf(CYAN "Jogador %d: No banco - Esperando\n" RESET, id);
    sleep(2);
    system("clear");
    pthread_cond_wait(&jogador_cond, &lock); //espera ate ter espaco em quadra
    no_banco--;
}

em_quadra++;
```

```
printf(RED "Jogador %d: Cansei - Vou descansar\n" RESET, id);
pthread_cond_signal(&juiz_cond);
pthread_cond_signal(&estrela_cond);
pthread_cond_signal(&jogador_cond); //libera espaco na quadra para o jogador
```

Para o thread da estrela o processo é muito parecido, mas como ela possui prioridade sobre o jogador comum, ele não tem que verificar se existem jogadores esperando para entrar.

Depois que isso é feito, os jogadores devem esperar ter 5 jogadores em quadra, quando isso acontece, o juiz é acordado para que a partida possa ser iniciada, como é possível visualizar abaixo, um jogador é responsável por avisar o juiz que os 5 jogadores estao prontos, e assim ele, o juiz, começar a partida.

```
pthread_mutex_lock(&lock);
if(em_quadra == 5 && start == 1) pthread_cond_signal(&juiz_cond);
pthread_mutex_unlock(&lock);

pthread_mutex_lock(&lock);
pthread_cond_wait(&comecar_jogo_cond, &lock); //espera o juiz apitar o comeco
    do jogo
pthread_mutex_unlock(&lock);
```

```
void * Juiz(void * data){
    while(1){
        pthread_mutex_lock(&lock);
        pthread_cond_wait(&juiz_cond, &lock);

        if(start){
            print_dados();
            printf(GREEN "-> Juiz: Temos 5 jogadores em quadra - Apita o apito
                - Comeco!\n" RESET);
            sleep(5);
            start = 0;
        }
        else {
            printf(CYAN "-> Juiz: Susbtituicao - Apita o apito - Volta jogo\n"
                RESET);
            sleep(5);
            system("clear");

        }

        pthread_cond_broadcast(&comecar_jogo_cond); // libera todos os
            jogadores para o jogo
```

Das variaveis acima as que mais devem ser chamadas atenção é a **start**, ela é inciada globalmente como 1, e depois que a partida é inciada é atribuido o valor 0 para ela, já que uma partida não começa duas vezes e a variavel de condição **comecar_jogo_cond**, pois cada jogador quando entra em quadra deve esperar esse sinal que enviado pelo juiz para que o jogo só aconteça enquanto estiverem 5 jogadores em quadra.

Depois que uma partida é inciada e os jogadores ficam jogando por um tempo aleatório, depois desse tempo eles ficam cansados e pedem para sair da partida, abrindo espaço na quadra para que outro jogador posssa entrar. Ou seja, o jogador que está cansado manda um sinal para os jogadores que estão esperando para que eles ”acordem”e entrem na partida para jogar, como é possível visualizar no trecho de código abaixo:

```
printf(RED "**Estrela** %d: Cansei - Vou descansar\n" RESET, id);
pthread_cond_signal(&juiz_cond);

pthread_cond_signal(&estrela_cond);
pthread_cond_signal(&jogador_cond);
pthread_mutex_unlock(&lock);
```

Além disso, existe também o sinal para o juiz, **juiz_cond** que avisa ao juiz que um jogador vai sair então haverá uma substituição de jogadores fazendo com que ele, o juiz, verifique

novamente que os 5 jogadores estão em quadra e assim proceda com a partida.

Sobre detalhes mais técnicos para o desenvolvimento do algoritmo, temos o uso de apenas um lock, que é utilizado para que cada pessoa do problema faça sua tarefa com exclusão multua, ou seja, apenas um jogador pode entrar na partida ou ser substituido por vez, ou seja, o acesso as variaveis que controlam quem está na partida ou não são feitos com exclusão múltua. Porém todas as threads que estão em quadra estão jogando até cansarem, assim como as que estão descansando também estão simultaneamente ficando preparadas para voltar a partida, tornando assim o problema e sua solução concorrente.

```
pthread_cond_t juiz_cond = PTHREAD_COND_INITIALIZER;
pthread_cond_t estrela_cond = PTHREAD_COND_INITIALIZER;
pthread_cond_t jogador_cond = PTHREAD_COND_INITIALIZER;
pthread_cond_t comecar_jogo_cond = PTHREAD_COND_INITIALIZER;
```

Também existem quatro variaveis de condição como visto acima, sendo responsáveis por manter a prioridade das estrelas sobe os jogadores comuns, trabalhar com o juiz e fazer com que o jogo só comece quando estiverem cinco jogadores em quadra. Outro detalhe é que foram utilizadas váriaveis globais para que fosse possível visualizar e entender melhor a situação que está ocorrendo, mostrando dados úteis ao usuário através do terminal.

4. Conclusão

A partir deste trabalho foi possível ter uma visão melhor sobre a Programação Concorrente, já que foi proposto a criação de um problema que possa ser resolvido com este paradigma, fazendo com que tivessemos que ser capazes de pensar num problema e imeditamente pensar em como o resolveriamos desta forma.

As principais dificuldades se encontram na hora de debugar um problema realizado utilizando o paradigma concorrente, já que como existem muitas threads realizando a mesma ativadade, fica dificil identificar onde acontecem os famosos deadlocks, starvations e outros problemas comuns que acompanham este paradigma de programação.

No meu problema em especial tive dificuldade em identificar um problema que era, os jogadores mandavam um sinal para o juiz acorda, porém o juiz ainda não tinha entrado em espera, ou seja, quando o juiz entrava em espera já não existiam jogadores capazes de acorda-lo, fazendo com que ele ficasse em deadlock. Para resolver isso, a thread do juiz deve ser a primeira a ser criada e com uma variável flag fui capaz de solucionar o problema.

Com isso este trabalho se mostrou bastante proveitoso para espandirmos nosso conhecimento e aumentar o desenvolvimento dentro da disciplina, já que foi colocado em pratica muitos dos conhecimentos adquiridos em sala de aula. O problema do Time de Basquete foi resolvido com sucesso e qualidade esperada e pessoalmente estou satifesito com o resultado do trabalho proposto.

5. Referencias

Material de aula do professor Eduardo Alchieri. https://cic.unb.br/~alchieri/disciplinas/graduacao/pc/pc.html

Manual da POSIX Threads. https://computing.llnl.gov/tutorials/pthreads/