Relátorio

Dupla: Kauan Oliveira Freitas; Thays da Silva Mariano

1. Introdução

O projeto “Adivinhe a Moeda” tem como objetivo demonstrar, de forma prática, coordenação e concorrência de tarefas em sistemas operacionais, utilizando threads POSIX, semáforos contáveis e mutexes. O jogo simula um servidor concorrente onde múltiplos jogadores disputam moedas limitadas, tentando adivinhar o número secreto de cada uma.

2. Estrutura Geral do Sistema

O sistema segue o modelo cliente-servidor:

- O servidor em C gerencia as moedas e o controle de concorrência.
- Os clientes (via navegador ou curl) interagem com endpoints como /collect, /guess, /status e /reset.

2.1 Threads

Cada requisição recebida pelo servidor é tratada em uma thread independente. Além disso, para cada moeda coletada, é criada uma thread temporizadora responsável por aguardar o tempo limite que o cliente tem para acertar o valor da moeda coletada. Caso a moeda não seja descoberta é chamado um *sem_post()*. Isso demonstra execução paralela e assíncrona, simulando múltiplos jogadores disputando com o mesmo conjunto de recursos.

2.2 Semáforo

O semáforo representa o número de moedas disponíveis para coleta. Ele controla quantas threads podem acessar o recurso por vez. A operação *sem_wait()* decrementa o semáforo para avisar que um dos recursos que antes estava disponível, já não se encontra mais acessível. Já a operação *sem_post()* incrementa o semáforo, avisando que um novo recurso está livre para ser utilizado.

2.3 Mutex

O mutex é utilizado para bloquear os recursos que já estão sendo acessados por alguma thread cliente, para dessa forma evitar que ocorra inconsistências nos recursos globais do sistema. Sem ele, duas threads poderiam alterar o estado da mesma moeda ao mesmo tempo, resultando em inconsistências.

3. Adendo Criativo

Como adendo criativo, o projeto base foi modificado para incluir a funcionalidade de identificação do jogador e um placar. Essa adição torna o jogo mais interativo e competitivo, além de introduzir um grau adicional de complexidade na coordenação de tarefas, por exigir o gerenciamento seguro de um novo recurso global.

3.1 Implementação

Para implementar essa funcionalidade foi criada uma estrutura global (leaderboard) responsável por armazenar o nome do jogador e sua pontuação. Outro mutex foi utilizado para proteger o acesso a essa estrutura, garantindo que apenas uma thread possa modificá-la por vez, evitando inconsistências.

Além disso, os endpoints originais /collect e /guess foram modificados para receber o nome do jogador como parâmetro. Essa mudança permite ao servidor identificar qual jogador realizou determinada ação e validar se o palpite é feito pelo mesmo usuário que coletou a moeda.

4. Conclusão

O projeto demonstra, de forma prática e visual, conceitos fundamentais de coordenação entre tarefas. O semáforo controla o número de recursos disponíveis, o mutex garante acesso exclusivo de uma única thread por recurso e as threads permitem execução concorrente. O jogo possibilita observar bloqueios reais de threads e sincronização entre múltiplas entidades, conectando teoria e prática dos mecanismos POSIX.

Resumo dos mecanismos utilizados:

Mecanismo	Função no projeto	Conceito representado
Semáforo	Controla número de moedas disponíveis	Controle de acesso a recursos finitos
Mutex	Protege alterações no vetor coins[]	Protege recursos
Threads	Permitem múltiplos jogadores e timers paralelos	Execução concorrente e bloqueio