Sistemas Operacionais I - SSC0140

Prof.ª Kalinka Regina Lucas Jaquie Castelo Branco

Anlee Feng Chang - 12563690

Giovanna Pedrino Belasco - 12543287

Laura Fernandes Camargos - 13692334

Vitor Laperriere de Faria– 12689085

Sumário

Descrição do jogo

O jogo da Forca desenvolvido pelo grupo é uma adaptação do clássico jogo da Forca, apresentando uma abordagem multijogador envolvendo um jogador principal e dois jogadores secundários, sendo ambos representados por threads. Ao iniciar o jogo, uma mensagem de boas-vindas é exibida, acompanhada de uma breve explicação das regras.

O jogador principal tem a tarefa de adivinhar a palavra inserindo letras. Cada letra correta revela sua posição na palavra oculta, e cada letra errada reduz o número de tentativas disponíveis. Os jogadores secundários tentam adivinhar letras aleatórias em intervalos de tempo variáveis.

O desafio é adivinhar uma palavra escolhida aleatoriamente a partir de um arquivo de palavras. A cada rodada, cada letra correta revela sua posição na palavra oculta, enquanto letras erradas reduzem o número de tentativas restantes.
O jogo continua até que a palavra seja totalmente adivinhada ou o número máximo de tentativas seja atingido.

Ao final do jogo, mensagens distintas são exibidas com base no resultado:

- Se o jogador principal adivinhar a última letra e consequentemente a palavra, ele vence a partida.
- Se um jogador secundário adivinha a última letra e consequentemente a palavra, as threads vencem a partida.
- Se o número máximo de tentativas é atingido sem sucesso, uma mensagem de derrota é exibida.

O jogo desenvolvido utiliza programação multithread, semáforos e manipulação de arquivos para criar uma experiência interativa de jogo da Forca na linguagem C++.

Utilização de threads e semáforos

No contexto do jogo, a integração de semáforos desempenha um papel crucial na gestão da concorrência entre as threads e na prevenção de conflitos durante as jogadas. Dois semáforos distintos são empregados, cada um com uma função específica que contribui para a sincronização eficiente do jogo.

Controle de Tentativas:

O semáforo mutex (*sem_t mutex*) desempenha um papel crucial no controle de tentativas no jogo da forca. Ele atua como um guardião da variável global "tentativa", garantindo que apenas uma thread por vez tenha a permissão necessária para fazer uma tentativa. Isso cria uma região crítica, onde o acesso à "tentativa" é exclusivo, evitando condições de corrida. Quando uma thread obtém a permissão do semáforo, ela pode validar sua tentativa de forma segura. Esse mecanismo assegura que a manipulação da "tentativa" seja feita de maneira ordenada e sem conflitos, contribuindo para a consistência do jogo.

Sincronização de Rodadas:

O outro semáforo (*sem_t sem_inicio_nova_rodada*) desempenha um papel crucial na sincronização das rodadas do jogo. Ao ser inicializado com valor 0 - *sem_init(&sem_inicio_nova_rodada, 0, 0)* - ele atua como uma barreira inicial e cada thread que termina sua inicialização sinaliza o semáforo, incrementando seu valor. Essa sincronização é vital para evitar que threads executem ações em rodadas distintas, o que poderia resultar em inconsistências no estado do jogo. Após cada rodada, o semáforo é incrementado para permitir que as threads prossigam, garantindo que apenas uma thread por vez entre na região crítica onde ocorre a tentativa de adivinhar a letra durante uma rodada.

Uso de Threads na Interatividade do Jogo:

A implementação de threads é fundamental para criar uma experiência de jogo interativa e responsiva. As threads são empregadas para representar os **jogadores e suas ações independentes**. Cada thread modela um jogador, seja o principal ou secundário, e executa

suas jogadas de maneira concorrente. Isso possibilita a execução simultânea de diferentes partes do código, contribuindo para a dinâmica do jogo.

JogadorUsuario: A função jogadorUsuario representa a interação do jogador principal, onde o código da thread está concentrado na espera da entrada do usuário para a tentativa de adivinhar uma letra. Essa thread é referenciada no início do jogo, quando a função jogadorUsuario é passada como argumento para a criação da primeira thread usando ***pthread_create(&jogadores[0], NULL, jogadorUsuario, NULL)***. Essa chamada inicia a execução da thread do jogador principal, responsável por interagir com o usuário, capturar a entrada e realizar as tentativas de adivinhação.

JogadorThread: A função jogadorThread encapsula o comportamento dos jogadores automáticos (threads secundárias). Cada thread secundária espera aleatoriamente antes de tentar entrar na região crítica, simulando uma dinâmica mais autônoma. As threads secundárias são referenciadas em um loop, onde duas threads são criadas utilizando a função *pthread_create* com a função jogadorThread como argumento usando ***pthread_create(&jogadores[i], NULL, jogadorThread, &n_dos_jogadores[i])***, onde ***&n_dos_jogadores[i]*** é passado para identificar cada jogador secundário de forma única. Essa estratégia de passar um identificador é útil para distinguir o comportamento de cada jogador secundário no contexto do jogo.

Como instalar e rodar

Bibliotecas e especificações:

É importante destacar que as bibliotecas e funções empregadas no código fornecido são otimizadas para sistemas Linux ou ambientes Unix-like, logo para garantir o funcionamento adequado do programa, é necessário utilizar o sistema operacional Linux.

Em caso de execução do jogo em um ambiente Windows, algumas adaptações podem ser necessárias devido às discrepâncias nas bibliotecas e chamadas de sistema entre os sistemas operacionais.

Bibliotecas utilizadas:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
```

Requisitos Mínimos:

Certifique-se de estar utilizando um sistema operacional compatível com o Make. O Makefile fornecido no arquivo zipado foi projetado para sistemas Unix/Linux, mas também pode ser executado em sistemas Windows com o GNU Make ou o MinGW. Como o jogo foi desenvolvido em C++, para executá-lo é necessário ter instalado em seu computador o compilador “g++”.

Passo a passo para executar o Makefile:

Para iniciar a execução do jogo, é necessário abrir o terminal no diretório onde está localizado o Makefile e os arquivos do projeto.

Feito isso, execute o comando “make” ou “make all” para compilar o projeto.

Para executar o programa, execute o comando “make run”.
Com o jogo iniciado, você verá a tela de menu printada no próprio terminal que possui algumas instruções sobre como o jogo funciona.

Link para o vídeo de explicação:

https://drive.google.com/file/d/1PDgvDLshXEaLechK9KjW6fLL3CeCrxkB/view?usp=sharing