UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
LABORATÓRIO DE SISTEMAS OPERACIONAIS
MARCOS TULIO AMARIS GONZALEZ.

PROJETO FINAL - Snake Game

TEMA

Daniel Naiff da Costa
Gabriel Lemos da Silva Mastub
Raiane Gabriele Brito de Oliveira
Rogério dos Anjos Barbosa

Sobre o jogo:

No projeto final da disciplina ”Laboratório de Sistemas Operacionais”, fomos desafiados com a seguinte proposta: Um jogo ou aplicativo que utilize threads e semáforos. Pensando nisso, seria importante verificar a usabilidade desses dois componentes, ou seja, criar um “tráfego” dentro do projeto que seria coordenado pelos requisitos acima citados. O jogo da cobra, ou snake game, é muito famoso e lida justamente com essa ideia de permissões e proibições. Implementamos o clássico jogo da cobra utilizando linguagem C++, com as bibliotecas `ncurses` para interface gráfica em terminal e `semaphore` para controle de concorrência. O objetivo é gerenciar simultaneamente a movimentação da cobra e a leitura de entradas do usuário, permitindo que o jogo seja jogado em tempo real. A estrutura do projeto se baseia no uso de threads e semáforos para evitar conflitos de dados entre a atualização da posição da cobra e as mudanças de direção realizadas pelo usuário.

Funcionamento das Threads:

Primeiramente, duas threads são criadas para o controle do jogo. Uma thread é responsável pela movimentação da cobra, que ocorre em intervalos regulares e de forma independente. Ela verifica constantemente a direção da cobra e a move um passo de cada vez, verificando também colisões com as paredes ou com o próprio corpo da cobra. Em caso de colisão, o jogo exibe uma mensagem de fim. A segunda thread lida com a leitura das teclas pressionadas pelo usuário. Esta separação permite que o jogador altere a direção da cobra enquanto a movimentação ocorre de forma fluida, sem interrupções.

Funcionamento do Semáforo:

Para garantir a integridade dos dados ao acessar e modificar a direção da cobra e sua posição, é utilizado um semáforo. Este recurso sincroniza o acesso às variáveis compartilhadas, prevenindo que ambas as threads leiam ou modifiquem os dados ao mesmo tempo, o que poderia causar erros durante a execução. Cada thread espera pelo sinal do semáforo antes de fazer alterações e o libera assim que termina a operação, mantendo o jogo consistente e evitando colisões de dados.

Como um todo, este projeto oferece uma implementação funcional e otimizada do jogo da cobra com o uso eficiente de `threads` e `monitores`, através de semáforos, para um controle seguro e sincronizado dos recursos compartilhados. O uso do `ncurses` proporciona uma interface simples e direta, adequada para jogos em terminal.

A importância das Threads e Semáforos no projeto:

O uso de threads e semáforo foi essencial para o funcionamento correto deste jogo, permitindo que o movimento da cobra e a captura das entradas do jogador ocorressem em paralelo, sem interferência entre si. As threads possibilitaram uma execução simultânea de tarefas distintas: uma thread mantém a cobra em movimento contínuo na tela, enquanto a outra monitora constantemente as teclas pressionadas

pelo usuário para alterar a direção da cobra. Essa divisão é crucial para garantir a fluidez do jogo, pois a cobra precisa se mover em intervalos regulares enquanto o jogador tem liberdade para modificar a direção a qualquer momento.

A sincronização das threads é assegurada pelo uso do semáforo, que age como um “guardião” das variáveis compartilhadas, como a posição e direção da cobra. Sem o semáforo, haveria risco de uma condição de corrida, em que as threads tentariam acessar ou modificar a posição e a direção da cobra ao mesmo tempo. Isso poderia causar inconsistências, como mudanças incorretas de direção ou falhas na detecção de colisões, que comprometem a lógica do jogo e podem resultar em falhas e comportamentos imprevisíveis.

Sem o uso de threads, o jogo seria limitado a um sistema de controle sequencial, onde o movimento e a captura de teclas aconteceriam de forma alternada, ou seja, uma ação sempre teria que esperar a outra. Isso tornaria o jogo muito menos responsivo, já que o jogador teria que aguardar cada atualização do movimento para registrar uma nova direção. Dessa forma, a importância das threads e do semáforo é evidente: eles viabilizam um funcionamento responsivo e seguro, essencial para a experiência de jogo em tempo real.

Link do Repositório: https://github.com/Flaeury/Projeto-Final-LAB-SO
Link da Apresentação: https://www.canva.com/design/DAGVPz-sHIY/0-X6X78L6MRT4rRTEhe4HQ/edit?utm_content=DAGVPz-sHIY&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton