Documentação: Sistema de Desarmamento Concorrente

Desenvolvedores

- Nome: Mario Cesar - Matrícula: 2211250
- Nome: Marco Barem - Matrícula: 2221022

Introdução

Este projeto foi desenvolvido como parte da disciplina de **Programação Concorrente**. O objetivo principal é simular um sistema de desarmamento de bombas, onde múltiplas threads (representando "Tedax") interagem de forma concorrente com diferentes módulos explosivos. A implementação utiliza **linguagem C**, focando no uso de **threads**, **mutexes**, **variáveis de condição** e a biblioteca **ncurses** para sincronização e exibição gráfica.

Descrição do Jogo

O sistema simula um cenário de desarmamento de bombas. O jogador deve gerenciar vários "Tedax", que são designados para desarmar módulos explosivos. Cada módulo tem atributos como dificuldade e tempo de desarme, e há um número limitado de bancadas disponíveis.

Regras e Fluxo do Jogo

1. **Geração de Módulos**:
   - Os módulos são gerados automaticamente pela thread `thread_mural` em intervalos de tempo configuráveis.
   - Cada módulo possui um nome, dificuldade, tempo estimado para desarme e status.
2. **Designação Manual**:
   - A thread `thread_coordenador` permite que o jogador associe um módulo específico a um "Tedax" disponível.
3. **Desarmamento**:
   - Um "Tedax", ao receber um módulo, ocupa uma bancada e realiza o desarmamento.
   - Após a conclusão, a bancada é liberada, e o módulo é marcado como desarmado.
4. **Encerramento**:
   - O jogo termina quando todos os módulos são desarmados ou o tempo limite é atingido.

Estrutura do Projeto

Arquivos

1. **`main.c`**:
   - Gerencia o fluxo principal do programa, inicializa as threads e finaliza o jogo.
2. **`funcoes.c`**:
   - Contém as implementações das funções principais, como geração de módulos, inicialização do jogo e lógica das threads.
3. **`funcoes.h`**:
   - Define estruturas, constantes e protótipos de funções utilizadas no projeto.
4. **`makefile`**:
   - Facilita a compilação e limpeza do projeto.

Estruturas de Dados

1. **`Modulo`**:
   - Representa um módulo explosivo com atributos como nome, dificuldade, tempo de desarme, e status.
2. **`Tedax`**:
   - Representa um especialista que desarma módulos, com atributos como habilidade e status.
3. **`FilaEspera`**:
   - Gerencia filas de espera para acesso às bancadas.

Threads

1. **`thread_mural`**:
   - Gera módulos periodicamente e os adiciona à lista global.
2. **`thread_exibicao`**:
   - Atualiza a interface gráfica em tempo real, exibindo informações sobre módulos, Tedax e bancadas.
3. **`thread_coordenador`**:
   - Permite ao jogador designar módulos a Tedax.
4. **`thread_tedax`**:
   - Cada Tedax é controlado por uma thread, que executa o desarmamento de módulos designados.

Sincronização

- **Mutexes**:
  - Protegem seções críticas, como a lista de módulos e o status dos Tedax.
- **Variáveis de Condição**:
  - Garantem que threads aguardem por eventos, como a designação de módulos.

Compilação e Execução

Requisitos

- Compilador GCC
- Biblioteca **ncurses**

```
bash
sudo apt install libncurses5-dev libncursesw5-dev
```

Comandos

- **Compilar**:

```
Bash
Make clean
make
```

- **Executar**:

```
bash
./jogo
```

- **Limpar Arquivos de Compilação**:

```
bash
make clean
```

Conclusão

Este projeto implementa conceitos de **programação concorrente**, como sincronização e controle de recursos compartilhados. A interface visual, criada com **ncurses**, proporciona uma experiência interativa e imersiva. O uso de threads possibilita a execução paralela e demonstra a importância de técnicas de sincronização, como mutexes e variáveis de condição.

Repositório GitHub: https://github.com/marionobrega/TrabalhoConcorrente

---