Keep Solving and Nobody Explodes - Versão de Treino

Documentação de Implementação - Programação Concorrente

Claudio da Aparecida Meireles Filho

Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa (IDP)
Ciência da Computação / Engenharia de Software
Disciplina: Programação Concorrente | Professor: Jeremias Moreira Gomes

2025/2

RA: 2321070

Resumo. Simulador em C que utiliza threads POSIX para coordenar Tedax virtuais na desativação de módulos de uma bomba. O projeto destaca sincronização por mutex e variáveis de condição, evitando race conditions e busy waiting, além de explorar configurações assimétricas entre Tedax e bancadas.

Abstract. A C simulator that uses POSIX threads to coordinate virtual bomb disposal experts (Tedax) in defusing bomb modules. The project highlights synchronization through mutex and condition variables, avoiding race conditions and busy waiting, while exploring asymmetric configurations between Tedax units and workbenches.

1. Arquitetura de Threads

Executam-se entre 5 e 8 threads simultaneamente, conforme descrito na Tabela 1.

Tabela 1. Threads do sistema e suas funções

Thread	Qtd	Função
Main (Coordenador)	1	Processa entrada e monta comandos
Mural de Módulos	1	Gera módulos aleatórios periodicamente
Timer	1	Decrementa tempo e sinaliza fim
Display	1	Atualiza interface ncurses a cada 100ms
Tedax	1-3	Desarmam módulos nas bancadas

As threads compartilham fila de módulos, bancadas, estado do jogo e buffer de entrada, exigindo mecanismos de sincronização adequados para evitar condições de corrida.

2. Seções Críticas e Sincronização

A Tabela 2 apresenta os recursos compartilhados e seus mecanismos de proteção.

Tabela 2. Recursos compartilhados e mecanismos de sincronização

Recurso	Mutex	Condição	Threads
Fila de Módulos	fila.mutex	cond_nao_vazia	Mural, Tedax, Main
Bancadas	bancada[i].mutex	cond_livre	Tedax

Recurso	Mutex	Condição	Threads
Estado do Jogo	mutex_estado	cond_fim_jogo	Todas
Buffer de Comando	mutex_comando	-	Main, Display

2.1. Protocolo das Bancadas

O protocolo de acesso às bancadas segue três etapas: (1) pthread_mutex_lock protege o teste de disponibilidade; (2) ao ocupar, registra estado e tedax_id; (3) ao liberar, pthread_cond_broadcast(&cond_livre) desperta Tedax aguardando.

2.2. Assimetria (Bônus)

Se houver mais Tedax que bancadas, cada Tedax aguarda em cond_livre com timeout controlado, evitando busy waiting e garantindo fila justa.

3. Fluxo de Execução

3.1. Ciclo do Tedax

O ciclo do Tedax compreende: (1) aguardar tarefa em cond_tarefa; (2) receber módulo e bancada designada; (3) tentar ocupar bancada, aguardando sinalização se ocupada; (4) simular resolução, verificar instrução e atualizar estatísticas protegidas por mutex_estado; (5) liberar bancada e retornar ao passo 1.

3.2. Processamento de Comandos (Main)

A thread principal: (1) lê buffer com mutex_comando e valida formato [TEDAX][TIPO][BANCADA][INSTRUÇÃO]; (2) verifica Tedax disponível e busca módulo do tipo; (3) remove módulo da fila, atribui ao Tedax e sinaliza cond_tarefa; (4) mantém flags executando e resultado_final sob mutex_estado para encerramento consistente.

4. Prevenção de Problemas

A Tabela 3 apresenta os riscos identificados e suas respectivas mitigações.

Tabela 3. Riscos e estratégias de mitigação

Risco	Mitigação
Deadlock	Ordem previsível de locks; liberação garantida antes de esperar condições
Race Condition	Toda variável compartilhada protegida por mutex dedicado
Starvation	broadcast ao liberar bancadas e fila FIFO de módulos
Busy Waiting	Espera bloqueante em condições para mural, Tedax e bancadas

Flags de controle são atualizadas sob mutex_estado, e pthread_join sincroniza todas as threads antes de restaurar a tela ncurses.

5. Guia de Utilização

5.1. Compilar e Executar

Em Ubuntu 24.04, execute: sudo apt install build-essential libncurses5-dev, seguido de make e make run (ou ./bomb_defuser).

5.2. Como Jogar

Observe os módulos pendentes no formato [ID] TIPO: INSTRUÇÃO. Monte o comando [Tedax][Tipo][Bancada][Instrução] (exemplo: 1f1rgb) e envie com ENTER. Comandos especiais: p (pausar), h (ajuda), q (sair), Ctrl+C (encerramento forçado com exibição do motivo).

5.3. Tipos de Módulos

A Tabela 4 descreve os tipos de módulos disponíveis.

Tabela 4. Tipos de módulos e suas instruções

Tipo	Letra	Instrução
Fios	f	Cores: r, g, b, y (ex.: rgb)
Botão	b	Pressionamentos: p repetido (ex.: pppp)
Sequência	s	Números 1-4 (ex.: 1234)
Simon	i	Direções u, d, l, r (ex.: udlr)

5.4. Configurações Disponíveis

O jogo permite configurar: número de Tedax (1 a 3), número de bancadas (1 a 5), tempo da partida em segundos, meta de módulos para vencer (pode ser ignorada no modo infinito) e dificuldade (1-3) que ajusta a geração de módulos.

6. Conclusão

O projeto demonstra aplicação prática de programação concorrente em C: uso de múltiplas threads coordenadas, seções críticas protegidas, sincronização por mutex/condições e exploração de configurações assimétricas. A modularização em arquivos separados facilita manutenção e extensão para novos tipos de módulos ou políticas de agendamento.

Referências

Tanenbaum, A. S. (2010) "Sistemas Operacionais Modernos", 3ª ed., Pearson.

Barney, B. (2024) "POSIX Threads Programming", Lawrence Livermore National Laboratory. https://computing.llnl.gov/tutorials/pthreads/

Dickey, T. (2024) "Ncurses Programming HOWTO", Linux Documentation Project. https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/