Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

Sistemas Operativos 23/24

Trabalho Prático – Programação em C para UNIX

Labirinto MultiJogador

Gonçalo Leite – 2018014643

Fernando Pereira - 2020154532

JogoUI

Visão Geral

O programa JogoUI atua como interface do utilizador para um jogo multiplayer, oferecendo uma experiência interativa aos jogadores. Utilizando a biblioteca ncurses em C, o JogoUI exibe o estado do jogo, recebe entradas do utilizador e comunica com o motor do jogo por meio de pipes, FIFOs e threads.

Estrutura do Código

Inclusões de Bibliotecas:

O código faz uso da biblioteca ncurses.h e outras bibliotecas padrão do C para manipulação de interfaces gráficas e operações de sistema.

Definições de Constantes

• LINHAS e COLUNAS: Representam as dimensões do mapa do jogo.

• MOTOR_LOCK_FILE: Nome do arquivo de bloqueio do motor (para apenas executar um motor de cada vez).

Estruturas de Dados

• Jogador: Estrutura que armazena informações sobre um jogador, como nome, caracter associado e coordenadas no mapa.

• Mensagem: Estrutura para representar mensagens trocadas entre jogadores.

Variáveis Globais

• jogadortemp: Representa o jogador atual.

• msg: Armazena informações sobre mensagens trocadas entre jogadores.

• mapa[LINHAS][COLUNAS]: Matriz que representa o mapa do jogo.

• jogadoresAtivos[MAX_PLAYERS]: Array de jogadores ativos.

• njogadores: Número de jogadores ativos.

Threads e Funções Principais

• trataTeclado: Thread responsável por capturar entradas do teclado, permitindo que os jogadores se movam e enviem comandos.

• recebeLista: Thread que atualiza a lista de jogadores ativos e exibe essa informação na interface.

• recebemsg: Thread que recebe mensagens de outros jogadores e exibe na janela do chat.

• recebeMapa: Thread que atualiza a representação gráfica do mapa do jogo na interface.

• enviaPosXY: Função que envia as coordenadas do jogador atual para o motor do jogo.

Funções Auxiliares e Comunicação

• enviamsg: Função que envia mensagens para jogadores específicos por meio de FIFOs.

• desenhaMapa: Função que desenha o mapa na interface gráfica, diferenciando entre a janela principal do jogo, a janela de informações dos jogadores e a janela de entrada de comandos.

• encerraJogoUI e winJogoUI: Funções que enviam sinais para encerrar o JogoUI e indicar a vitória, respectivamente.

• Pipes e FIFOs: Utiliza FIFOs para comunicação eficiente entre o JogoUI e o motor do jogo.

O JogoUI utiliza a biblioteca ncurses para criar uma interface gráfica textual. Oferece funcionalidades como movimentação no mapa, envio de mensagens, exibição de jogadores ativos e interação com comandos.

Motor

Visão Geral

O programa implementa um servidor para um jogo multiplayer, utilizando diversas tecnologias em C, como pipes, sinais, threads e FIFOs, para facilitar a comunicação entre o servidor e os jogadores. O servidor mantém um mapa, controla os jogadores, lida com os movimentos e executa bots que interagem com o ambiente do jogo.

Estrutura do Código

O programa inclui várias bibliotecas padrão do C, como stdio.h, pthread.h, stdlib.h, unistd.h, signal.h, entre outras. Essas bibliotecas fornecem funcionalidades fundamentais para manipulação de threads, sinais, entrada/saída e comunicação interprocessual.

Estruturas de Dados

• Jogador: Estrutura que representa um jogador no jogo, com informações como nome, pid, caractere associado e coordenadas (x, y) no mapa.

• Comando: Estrutura para armazenar comandos que são enfileirados entre threads.

Variáveis Globais

• jogadores: Array de jogadores, armazenando informações sobre os jogadores conectados.

• atualizaJogadores: Instância da estrutura Jogador para atualização de informações de jogadores.

• nomesJogadores: String para armazenar os nomes dos jogadores conectados.

• posX e posY: Arrays para armazenar as posições x e y dos blocos no mapa.

Constantes

• LINHAS e COLUNAS: Constantes que representam as dimensões do mapa.

• Diversas constantes para identificar pipes e FIFOs usados na comunicação.

Funções Principais

• main(): Função principal do programa, inicia as threads principais e lida com a execução geral do jogo.

• recebeJogadores(): Thread que recebe informações de novos jogadores e os adiciona à lista de jogadores.

• atualizaPosXY(): Thread que atualiza as posições dos jogadores no mapa.

• bot(): Thread que executa um bot que interage com o ambiente do jogo.

• moveBloqueios(): Thread que move os blocos no mapa.

• get_comandos(): Thread para receber comandos do console e executar ações correspondentes.

Funções Auxiliares

• enqueueComando() e dequeueComando(): Funções para enfileirar e desenfileirar comandos entre threads.

• enviarComandoMotor(): Envia comandos ao motor do jogo.

Pipes e FIFOs

• Utiliza pipes e FIFOs para comunicação entre threads e processos.

• O programa cria e utiliza vários FIFOs para transmitir informações entre diferentes partes do código.

Sinais

• Utiliza sinais para tratar eventos como a interrupção do jogo (SIGINT) e encerramento de jogadores (SIGUSR1).

Lógica do Jogo

1. Inicialização: O programa inicia criando FIFOs e pipes necessários e lê o mapa do arquivo "mapa.txt".

2. Recebimento de Jogadores: A thread recebeJogadores aguarda a chegada de jogadores, atualiza o estado do jogo e envia notificações para os jogadores existentes.

3. Atualização de Posições: A thread atualizaPosXY monitora as atualizações nas posições dos jogadores e reflete essas alterações no mapa.

4. Bot: A thread bot executa um bot que interage com o ambiente do jogo, movendo-se e bloqueando caminhos.

5. Movimento de Bloqueios: A thread moveBloqueios move os blocos no mapa periodicamente.

6. Console de Comandos: A thread get_comandos permite comandos a partir do console para listar jogadores, bots, adicionar/remover blocos, iniciar e encerrar o jogo.

7. Encerramento do Jogo: O jogo pode ser encerrado através do comando "end", que encerra o motor, avisa os jogadores e encerra as threads.

Threads

1. hread Principal (main()):

• Função: Inicia o programa e coordena a execução das outras threads.

Responsabilidades:

• Cria o mapa do jogo.

• Inicializa as threads principais.

• Gerencia a criação e exclusão do arquivo de trava (MOTOR_LOCK_FILE).

• Lida com a execução geral do jogo.

2. Thread recebeJogadores():

• Função: Responsável por receber informações de novos jogadores.

Responsabilidades:

• Aguarda a chegada de jogadores através do FIFO PIPE_MOTOR.

• Atualiza o estado do jogo com as informações recebidas.

• Adiciona novos jogadores à lista de jogadores.

• Remove jogadores quando necessário.

3. Thread atualizaPosXY():

• Função: Atualiza as posições dos jogadores no mapa.

Responsabilidades:

• Monitora as atualizações nas posições dos jogadores recebidas através do FIFOPIPE_ATUALIZA.

• Atualiza o mapa refletindo as novas posições dos jogadores.

4. Thread bot():

• Função: Executa um bot que interage com o ambiente do jogo.

Responsabilidades:

• Cria um processo filho que executa o bot.

• Lê as mensagens do bot através de um pipe.

• Atualiza o mapa com as ações do bot.

5. Thread moveBloqueios():

• Função: Move os blocos no mapa periodicamente.

• Responsabilidades: Periodicamente, altera a posição dos blocos no mapa.

6. Thread get_comandos():

• Função: Recebe comandos do console e executa as ações correspondentes.

Responsabilidades:

• Aguarda comandos do console.

• Interpreta e executa os comandos, como listar jogadores, bots, adicionar/remover blocos, iniciar e encerrar o jogo.

7. Thread thread_comandos:

• Função: Thread criada para a função get_comandos.

• Responsabilidades: Facilita a execução concorrente da função get_comandos.

8. Thread thread_bot:

• Função: Thread criada para a função bot.

• Responsabilidades: Facilita a execução concorrente da função bot.

9. Thread thread_bloqueios:

•Função: Thread criada para a função moveBloqueios.

•Responsabilidades: Facilita a execução concorrente da função moveBloqueios.