Ano Letivo: 2023/2024

Relatório

SO

Trabalho realizado por:

Martim Alexandre Vieira Antunes

nº: 2022141890 **Curso:** LEI

Pedro Lino Neves Faneca

nº: 2022134142 **Curso:** LEI

Índice:

1. **Introdução**
2. **Estruturas de Dados**
   - **2.1. dataMsg**
   - **2.2. Jogador**
   - **2.3. Comando**
   - **2.4. Bloqueio**
   - **2.5. AtualizacaoJogadorA**
3. **Variáveis Globais**
   - **3.1. Duração do Período de Login (INSCRICAO)**
   - **3.2. Número Mínimo de Jogadores Necessários (NPLAYERS)**
   - **3.3. Duração do Jogo (DURACAO)**
4. **Named Pipes**
   - **4.1. Utilização das Named Pipes**
   - **4.2. Login dos Jogadores**
   - **4.3. Comunicação Contínua entre Motor e JogoUI do Mapa**
   - **4.4. Controlo da Deslocação dos Jogadores pelo Mapa**
5. **Threads**
   - **5.1. No JogoUI (Cliente)**
     - **5.1.1. Tratamento do Teclado**
     - **5.1.2. Entrega Contínua do Mapa**
   - **5.2. No Motor (Servidor)**
     - **5.2.1. Envio Contínuo do Mapa para JogoUI**
     - **5.2.2. Temporizador para Bloqueio das Pedras**
     - **5.2.3. Temporizador Geral do Jogo**
     - **5.2.4. Thread de Login**
6. **Sinais:**
   - **6.1. Utilização de Sinal SIGINT para o Término do Jogo**
7. **Conclusão**

1. Introdução:

No âmbito da Unidade Curricular de Sistemas Operativos, foi apresentado aos estudantes o desafio de desenvolver um jogo de labirinto em modo multijogador, onde todos os participantes estão conectados à mesma máquina UNIX.

Este relatório concentra-se exclusivamente na abordagem do grupo à segunda meta do projeto.A validação e os testes realizados durante esta fase já incorporam a presença de um servidor e de um ou mais clientes, proporcionando uma visão abrangente da operacionalidade do jogo em modo multijogador. Este relatório explora as escolhas arquiteturais e estratégias implementadas para a gestão da comunicação entre os jogadores, utilizando *named pipes* e *threads* como componentes centrais dessa abordagem.

2. Estruturas de Dados:

Estruturas Relevantes:

2.1. dataMsg:

Estrutura responsável por armazenar informações de mensagens trocadas entre o motor e o JogoUI, contendo dados como PID, mapa, identificador da mensagem, nome associado e informações adicionais.

- pid: Identificador de processo (PID) associado à mensagem.
- mapa: Matriz representando o mapa do jogo.
- idMsg: Identificador único da mensagem.
- nome: Nome associado à entidade (jogador ou sistema).
- letra: Caractere associado à entidade.

- aviso: Mensagem informativa ou de estado.

2.2. Jogador:

Representa um jogador no contexto do jogo, mantendo informações como caractere associado, nome, PID, identificador de mensagem, estado do mapa, tecla pressionada, mensagem informativa e uma flag de ocupação.

- letra: Caractere associado ao jogador no mapa.
- nome: Nome do jogador.
- pid: Identificador de processo (PID) do jogador.
- idMsg: Identificador único da mensagem associada ao jogador.
- mapa: Matriz representando o mapa do jogo.
- tecla: Tecla pressionada pelo jogador.
- aviso: Mensagem informativa ou de estado.
- ocupado: Flag indicando se a entrada está ocupada.

2.3. Comando:

Definição de comandos utilizados para interação no jogo, incluindo nome, descrição, função associada à execução e o número de argumentos esperados pela função.

- nome: Nome do comando.
- descricao: Descrição do comando.
- func: Função associada à execução do comando.
- argumentos: Número de argumentos esperados pela função do comando.

2.4. Bloqueio:

Estrutura que é responsável pela gestão as pedras no jogo.

- X: Coordenada x da pedra no mapa.
- Y: Coordenada y da pedra no mapa.
- DuracaoBloqueio: Tempo (em segundos) que a pedra permanece nas coordenadas.
-

2.5. AtualizacaoJogadorA:

Introduzida para comunicar ao motor do jogo as ações dos jogadores. A estrutura contém a direção na qual o jogador deseja se mover (cima, baixo, esquerda, direita) e o ID associado ao jogador.

- Direcao: direcao que o jogador premiu para se movimentar no mapa (inteiro de 0 a 3).
- Id: ID associado ao jogador que esta a tentar se mover.

3. Variáveis Globais:

No contexto deste trabalho, destacamos três variáveis globais cruciais que moldam a experiência de jogo:

3.1. Duração do Período de Login (INSCRICAO):

- Esta variável define o tempo, em segundos, durante o qual os jogadores têm a oportunidade de realizar o processo de login. Após o término desse período, as inscrições são encerradas, preparando-se para o início do jogo. Ajustar essa variável impacta diretamente a janela de tempo disponível para os jogadores se registarem no sistema.

3.2. Número Mínimo de Jogadores Necessários (NPLAYERS):

- Essa variável estipula o número mínimo de jogadores necessários para iniciar uma partida. O sistema aguarda até que, pelo menos, o número especificado de jogadores esteja inscrito antes de iniciar o jogo automaticamente.

3.3. Duração do Jogo (DURACAO):

- A variável DURACAO determina o tempo total, em segundos, que os jogadores têm para concluir o jogo com sucesso. Se o tempo se esgotar antes que um jogador ou equipe alcance a vitória, todos os jogadores perdem.

3. Named Pipes:

3.1 Utilização das named pipes:

Durante a implementação do projeto, foram concretizadas diversas funcionalidades que moldam a interação entre o motor (servidor) e o JogoUI (cliente), estabelecendo a base para a experiência de jogo. As principais realizações incluem:

3.1.1. Login dos Jogadores:

- Para viabilizar a participação dos jogadores, foi implementado um processo de login. Nesse contexto, é aberto um **named pipe** específico para a comunicação entre o JogoUI (cliente) e o motor (servidor), chamado `SERVER_FIFO`. Esse canal de comunicação é utilizado para realizar o login dos

jogadores, permitindo que se inscrevam jogadores no jogo e participem da sessão em andamento.

3.1.2. Comunicação contínua entre Motor e JogoUI do mapa:

- Após a conclusão do login, o mesmo SERVER_FIFO torna-se crucial para a comunicação contínua entre o motor e o JogoUI. Esse canal estabelece uma ponte eficiente para a troca de informações vitais durante o jogo, como a transmissão do mapa atualizado do motor para o JogoUI. Essa atualização contínua garante que os jogadores tenham uma representação em tempo real do ambiente de jogo, permitindo uma experiência envolvente e dinâmica.
- Essa estrutura de comunicação bidirecional, implementada por meio do SERVER_FIFO, não apenas facilita a entrada dos jogadores no jogo, mas também proporciona uma interface eficaz para a transmissão de dados cruciais, como o estado do mapa. Essas funcionalidades fundamentais solidificam a base para uma experiência de jogo coesa e interativa.

3.1.3. Controlo da Deslocação dos Jogadores pelo mapa:

- Para possibilitar a interação dos jogadores com o ambiente do jogo, foi introduzida a comunicação por meio da **named pipe** chamada SERVER_JOGO_FIFO. Esta pipe atua como um canal bidirecional de comunicação, permitindo que o JogoUI (cliente) transmita ao motor (servidor) a direção desejada pelo jogador no mapa. Por meio desta pipe, as informações sobre as direções escolhidas pelos jogadores são transmitidas, possibilitando a atualização imediata do estado do jogo no motor. Esse mecanismo proporciona uma experiência de jogo fluida e responsiva, onde as ações dos jogadores são refletidas em tempo real no ambiente virtual.

4.Threads:

4.1No JogoUI(Cliente):

Para garantir uma execução concorrente eficaz e uma experiência de jogo contínua, foram implementadas threads no lado do JogoUI. Essas threads desempenham papéis específicos, facilitando a interação do jogador com o ambiente virtual e a atualização constante do estado do jogo.

4.1.1. Thread `trataTecladoThread`:

- Responsável pela captura de inputs do teclado por parte do jogador no JogoUI. Essa thread utiliza a função `wgetch` para aguardar a entrada do jogador, permitindo a deteção de movimentos (setas direcionais) e comandos especiais (por exemplo, barra de espaço para inserir comandos). A execução concorrente dessa thread possibilita uma resposta instantânea aos inputs do jogador, contribuindo para a jogabilidade fluida.

4.1.2. Thread `recebeMapa`:

- Encarregada de receber constantemente as atualizações do mapa provenientes do motor (servidor). Essa thread utiliza a SERVER_FIFO para receber as informações sobre o estado atual do jogo. A atualização periódica do mapa permite que o JogoUI reflita em tempo real as mudanças ocorridas no ambiente do jogo. Essa comunicação assíncrona entre o motor e o JogoUI, mediada pela thread `recebeMapa`, é essencial para manter o jogador informado sobre o estado dinâmico do jogo.

Esta abordagem de threads no JogoUI oferece uma experiência de jogo interativa e envolvente, permitindo que o jogador influencie ativamente o curso do jogo enquanto recebe atualizações em tempo real do motor. A gestão eficiente dessas threads contribui para a fluidez e a responsividade do sistema como um todo.

4.2. No Motor (Servidor):

No lado do motor, ou seja, no servidor responsável por coordenar o jogo, a utilização de threads é crucial para garantir um ambiente de jogo dinâmico e responsivo. As threads desempenham funções específicas, desde a gestão do tempo até a comunicação constante com os clientes.

4.2.1. Thread `threadEnviaMapaParaJogoUI`:

- Constantemente envia atualizações do mapa para o JogoUI. Esta thread utiliza a SERVER_FIFO para transmitir as informações do estado do jogo para todos os jogadores conectados, possibilitando que eles visualizem em tempo real as mudanças no mapa do jogo.

4.2.2. Thread `temporizador_handler`:

- Faz a gestão do tempo de bloqueio das pedras do bot. Esta thread é responsável por controlar o tempo que as pedras permanecem no mapa antes de serem removidas.

4.2.3. Thread `temporizador_thread`:

- Responsável por controlar o tempo restante para o término do jogo. Define um limite de tempo durante o qual os jogadores devem alcançar seus objetivos. O uso dessa thread contribui para a implementação de uma dinâmica de jogo baseada em tempo, adicionando um elemento estratégico à experiência do jogador.

4.2.4. Thread `threadLogin`:

- Encarregada de lidar com os processos de login dos jogadores no início da execução do jogo. Essa thread gere a criação de novos jogadores, verificando os seus detalhes de login e atribuindo identificadores exclusivos. A existência dessa thread facilita a entrada de vários jogadores de forma simultânea e organizada.

A utilização coordenada destas threads no motor é essencial para criar um ambiente de jogo dinâmico, com eventos temporizados e comunicação constante com os jogadores, proporcionando uma experiência de jogo envolvente e estrategicamente desafiadora.

5.Sinais:

5.1. Utilização de Sinal `SIGINT` para o Término do Jogo:

A estratégia de sinalização foi empregada para possibilitar o término controlado do jogo, sendo o sinal `SIGINT` adotado lado do servidor (Motor). O servidor (Motor) responde ao sinal `SIGINT` através da função `termina`, que inicia o encerramento do servidor e notifica os clientes ocupados sobre o término iminente do jogo. Após essa ação, os pipes utilizados são removidos, e uma mensagem de despedida é apresentada antes do encerramento final do servidor.

6.Conclusão:

No decorrer deste projeto, desenvolvemos uma aplicação multijogador do jogo do labirinto, focada na interação entre o Motor (servidor) e os JogoUIs (clientes) por meio da utilização eficaz de *named pipes*. A estrutura de dados foi projetada para refletir de maneira precisa os elementos centrais do jogo, como jogadores, mensagens e bloqueios.
A introdução de *threads*, tanto no cliente quanto no servidor, proporcionou uma execução concorrente suave, enquanto a adoção de sinais, especialmente `SIGINT`, permitiu um encerramento controlado e ordenado do jogo, garantindo a liberação adequada de recursos.

A funcionalidade de login, juntamente com a transmissão eficiente de mapas, contribuiu para uma experiência de jogo interativa e envolvente. O uso extensivo de *named pipes* foi essencial para a comunicação eficaz entre os componentes do jogo, garantindo uma sincronização eficiente.