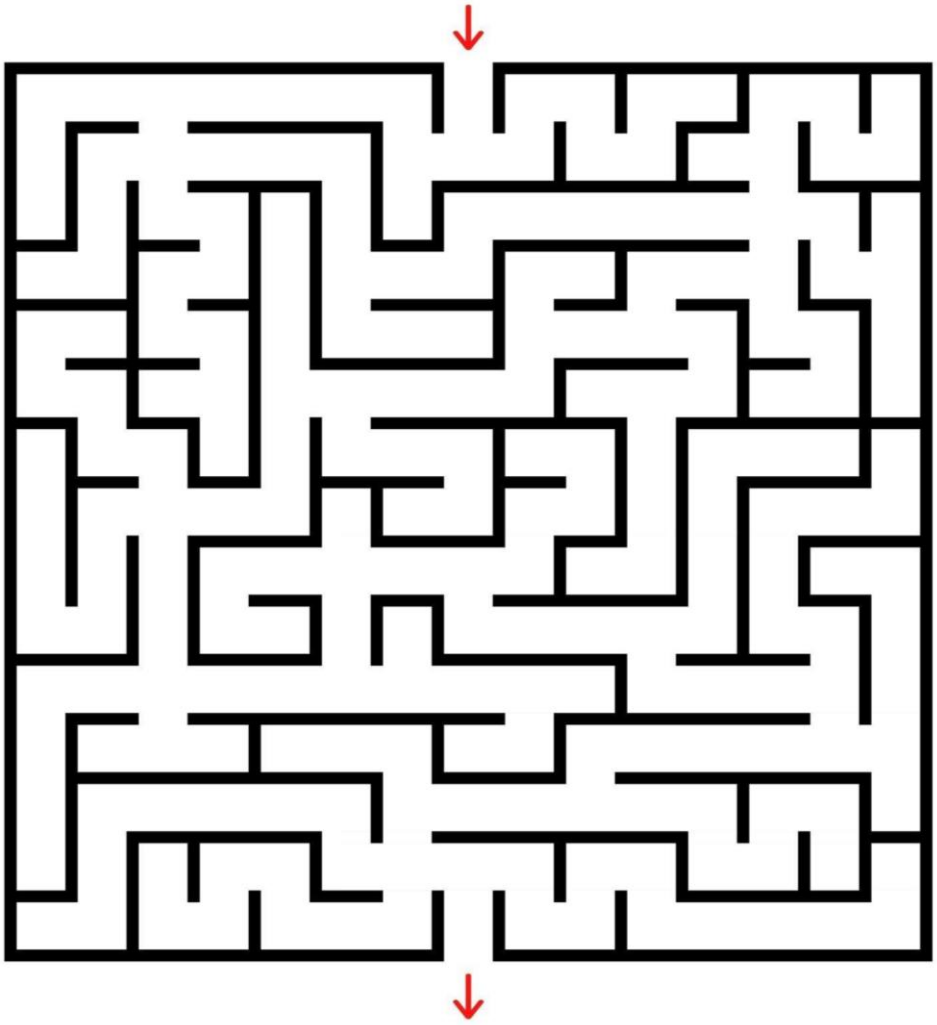
Sistemas Operativos 2023/24

Meta 1



Elementos do Grupo:

João Afonso Fonseca Costa Almas – 2021138417

Rodrigo Ramalho Ferreira – 2021139149

Índice

ÍNDICE 2
INTRODUÇÃO: 3
ESTRUTURA DO PROJETO: 3
Interface de Utilizador *3*
jogoUI_main.c: 3
jogoUI.c: 3
Motor do Jogo *3*
motor_main.c: 3
motor.c: 4
utils.c: 4
utils.h: 4
CONCLUSÃO: 4

Introdução:

O presente relatório descreve a estrutura do projeto em questão, que tem como objetivo a implementação de um jogo com interface de utilizador(*jogoUI*) e um *motor* subjacente. O projeto está organizado em diferentes partes, cada uma responsável por aspetos específicos do sistema.

Estrutura do Projeto:

Para este projeto optamos por dividir o projeto nas seguintes partes:

Interface de Utilizador

jogoUI_main.c:

- Função *main* do programa jogoUI.
- Ao tentarmos abrir o jogoUI, verificamos a existência do fifo do motor para determinar se o motor já está aberto. Se não estiver, o acesso ao jogo é negado e o programa é encerrado com uma mensagem de aviso.
- Após a validação do estado do motor, verificamos os argumentos do jogoUI. Se o número de argumentos não for válido, o acesso ao jogo também é negado.
- Em seguida, utilizamos a biblioteca ncurses.h para controlar a apresentação do jogo ao utilizador. Criamos quatro janelas: uma para a entrada de comandos do utilizador, outra para fornecer feedback sobre o que está acontecendo no jogo e os comandos executados, uma terceira que mostra os jogadores online no jogoUI e uma quarta que mostra o mapa do jogo.
- Além da criação dessas janelas, também validamos cada comando e os seus argumentos.

jogoUI.c:

- Aqui é onde criamos todas as funções utilizadas no JogoUI_main.c, por exemplo, a função executaComando_J, que faz uma cópia do comando com os argumentos introduzidos pelo utilizador usando a função strdup. A separação do comando em relação aos argumentos é feita através da função strtok, e com isso conseguir enviar diferentes mensagens com base no comando introduzido, dando assim feedback ao utilizador sobre o que está a acontecer.
- Também desenvolvemos um procedimento, inspirado no que o professor forneceu, que desenha a moldura de uma janela com base em sua altura, comprimento, linha e coluna inicial.

Motor do Jogo

motor_main.c:

- Função *main* do programa motor.
- Primeiramente verifica se já existe um ficheiro referente ao motor (FIFO_MOTOR)**:**
    - **Se existir** significa que o motor já está em execução e impede o processo de continuar.
    - **Se não existir** cria o FIFO_MOTOR de forma a impedir a execução de outros motores.

- De seguida é pedida a inserção de um comando e são validados os seus argumentos.
- Antes de terminar o processo, elimina o FIFO_MOTOR, indicando assim que o motor já não se encontra em execução.

motor.c:

- Contém todas as funções utilizadas no motor_main.c, como:
  - ***executaComando_M*** que funciona de maneira idêntica à função *executaComando_J* anteriormente descrita, mas que valida os comandos referentes ao motor.
  - ***GetFromBot*** que usa a função *fork* para gerar um processo descendente onde executa o programa do bot que gera as coordenadas e as envia para o primeiro processo através de um pipe anónimo.

utils.c:

- Ficheiro código fonte que armazena as funções que serão comuns aos programas *motor* e *jogoUI*, que para já é a função ***validaComando***, esta função verifica a existência do comando inserido e se o número de argumentos é válido.

utils.h:

- Neste ficheiro, além de conter o protótipo da função criada no *utils.c* e a constante responsável por definir o nome do fifo do *motor*, inclui também as estruturas de dados responsáveis por gerir as definições de funcionamento no *jogoUI* e no *motor*. Estas estruturas incluem a **PEDIDO**, que armazena informações do lado do *jogoUI* para que o motor possa aceder e tomar decisões com base nessas informações; a estrutura **USER**, que guarda informações do jogador, como o nome, o estado da conexão, se ativo ou não, e a posição do jogador no mapa; **PEDRAS**, onde são armazenadas as posições e a duração de cada pedra; **RESPOSTA**, que armazena uma mensagem e uma pedra enviada do *motor* para o *jogoUI*; e o **LABIRINTO**, onde são guardadas a lista de jogadores, o nível do jogo, o número de bloqueios, o número de pedras, e a lista de pedras existentes.

Conclusão:

Organizámos o projeto de forma a separar claramente a Interface do *jogoUI* e do *Motor* do Jogo, o que tornou a implementação mais organizada. A utilização de ferramentas como *ncurses.h* para a interface do utilizador foi essencial para uma melhor visualização da informação ao utilizador. Em suma, esperamos ter conseguido cumprir com todos os requisitos para esta meta.