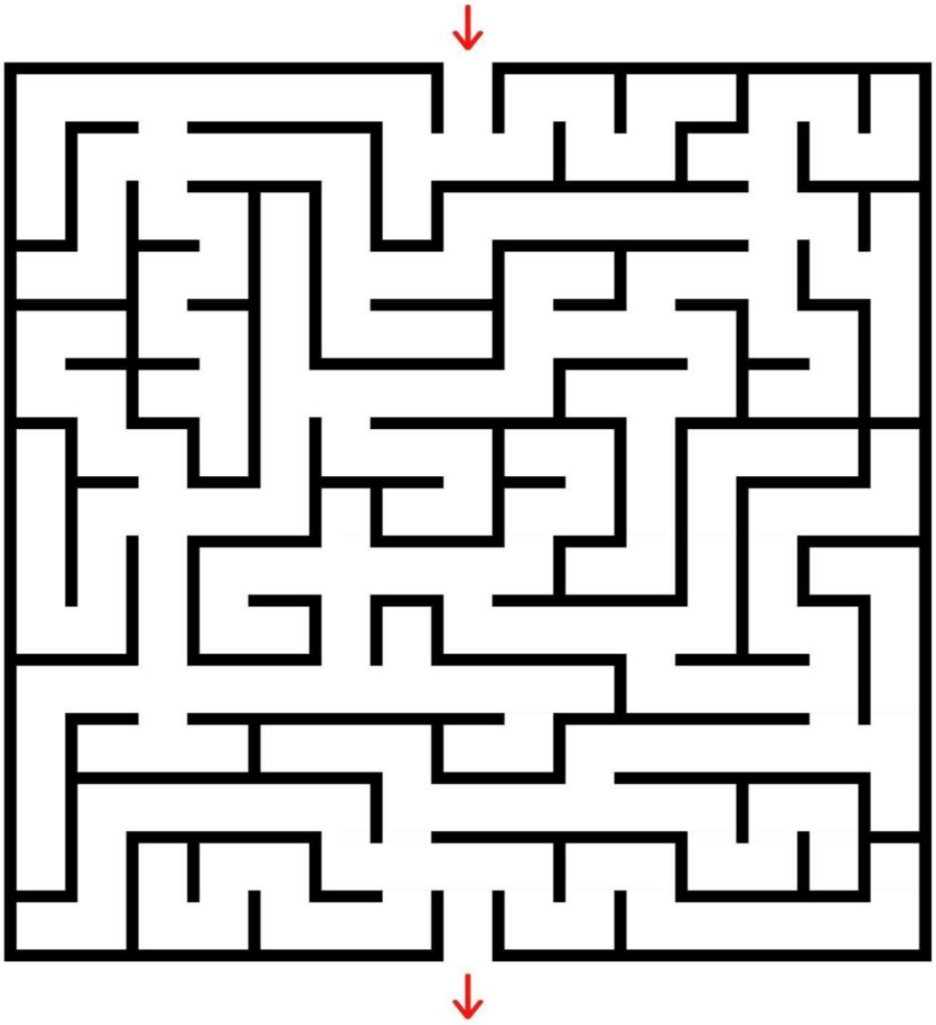
Sistemas Operativos 2023/24



Meta 2

Elementos do Grupo:

João Afonso Fonseca Costa Almas – 2021138417

Rodrigo Ramalho Ferreira – 2021139149

Índice

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 ou o nome do jogador já existir na lista de jogadores, o acesso ao jogo também é negado.
- Em seguida, utilizamos a biblioteca ncurses.h para controlar a apresentação do jogo ao utilizador. Criamos seis 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, uma quarta que mostra o mapa do jogo, uma quinta que mostra o nível que está atualmente, assim como a quantidade de blocos e pedras no mapa, e uma quinta janela que mostra o nome do jogador em questão e o tipo de cliente, se jogador ou espetador.
- Além da criação dessas janelas, utilizamos um select de maneira de conseguir digitar comandos e enviar dados para o motor enquanto recebemos informações por parte do motor.
- O jogoUI recebe sempre uma flag por parte do motor com um respetivo valor, e a partir dessa flag são lidos os respetivos dados e executadas as suas ações.

jogoUI.c:

- Aqui, optámos por criar todas as funções utilizadas no ficheiro JogoUI_main.c, incluindo aquelas relacionadas com os comandos, movimentação do mapa, preenchimento das estruturas, bem como funções responsáveis pelo envio e receção de informações para e do motor do jogo.
- 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 são criadas duas threads, a *comunicacaoCliente*, e a *timeInscricao*.
    - A thread *comunicacaoCliente* é inteiramente responsável pela comunicação do cliente. Esta thread recebe uma flag por parte do jogoUI e consoante essa flag é feito determinada ação, tais como, adicionar, após o login, o cliente como jogador ou espetador e enviar o jogo com mapa atualizado sempre que é feito alguma alteração por parte dos jogadores, por verificar a existência do destinatário da mensagem que o cliente quer enviar, por enviar a tabela de clientes no jogoUI, assim como remover um cliente da estrutura e enviar uma flag com o valor 7 para o jogoUI, que serve como confirmação que este pode sair.
    - A thread *timeInscricao* é responsável por decrementar o valor do tempo de inscrição, obtida através da variável de ambiente INSCRICAO. Assim que o tempo chegar a 0 e caso haja o número mínimo de jogadores o jogo então é enviado uma flag ao jogoUI para que o jogo seja iniciado. Caso não hajam jogadores necessários para a partida é enviada outra flag com valor -1 para que o jogo seja fechado e de seguida o motor também é fechado. Caso não hajam jogadores necessários para o jogo, mas é introduzido o comando begin no motor, então é atualizado a variável begin da estrutura TDATA, assim como o valor da inscrição passa a valor 0, para que os próximos clientes a entrar no jogoUI sejam do tipo espetador.
- Assim que é introduzido o comando end são terminadas as threads descritas acima, e libertados os recursos utilizados pelo programa, e elimina o FIFO_MOTOR, indicando assim que o motor já não se encontra em execução.

motor.c:

- Neste ficheiro, encontram-se todas as funções utilizadas no ficheiro motor_main.c. Estas incluem funções relacionadas com os comandos, os mapas de cada nível, os clientes (atualização, adição, remoção, envio para o JogoUI) e a verificação da existência de um cliente específico. Também estão presentes funções relacionadas com a inicialização dos dados das threads, a comunicaçãoCliente, a timeInscricao, a inicialização de variáveis de ambiente e outras funções essenciais para o correto funcionamento do programa.

utils.c:

- Ficheiro código fonte que armazena as funções que serão comuns aos programas *motor* e *jogoUI*, a função ***validaComando***, esta função verifica a existência do comando inserido e se o número de argumentos é válido. A função ***tolowerString*** e ***toupperString*** que recebem uma string e retornam essa mesmo string em minúsculas e maiúsculas, respetivamente.

utils.h:

- Neste ficheiro, além de conter o protótipo da função criada no *utils.c* também contém algumas constantes, tais como, a constante MAP_SIZE responsável por definir o tamanho do mapa, o N_LIN que contem o número de linhas do mapa, o N_COL que contem o valor de colunas do mapa, MAXUSERS que define o valor máximo de clientes na plataforma, PEDRASMAX que define o valor máximo de pedras, BLOCKMOVEIS que define o valor máximo de blocos e o FIFO_MOTOR que define o nome do fifo do *motor*.

structs.h:

- Este ficheiro contém todas as estruturas utilizadas pelos programas.
- Estas estruturas foram concebidas e desenvolvidas de forma cuidada. Criámos estruturas com o propósito de armazenar o conteúdo necessário para ser enviado entre o *motor* e o *jogoUI* e estruturas associadas a essas estruturas, com o intuito de funcionarem como “selo” do conteúdo da estrutura principal. Esta abordagem simplifica a interação entre o motor e a interface do jogo, pois basta verificar o valor do “selo”, que contém uma flag, para que o motor ou a interface do jogo saibam como lidar com a informação contida na estrutura.
- Estrutura PEDIDO_PLAYER que guarda o valor da flag, e o pid. Usado para pedir a listagem de jogadores ao motor.
- Estrutura PEDIDO_EXIT que guarda o valor da flag e o pid. Usada para enviar ao motor o “pedido” para sair do jogoUI.
- Estrutura CLIENT, que guarda informações do jogador, como o nome, o tipo de cliente, se jogador ou espetador, e a posição do jogador no mapa;
- A estrutura LOGIN, que contém informações associadas ao login de um cliente
- A estrutura F_LOGIN, serve como “selo” para estrutura LOGIN
- A estrutura MSG, recebe informações relativas a uma mensagem entre jogadores, tal como, o nome do remetente, do destinatário e o conteúdo da mensagem;
- A estrutura F_MSG, armazena a flag associada à estrutura MSG
- A estrutura MSG_REQUEST contém o nome do destinatário da mensagem
- F_MSG_REQUEST contém uma flag com valor associado à estrutura MSG_REQUEST

- A estrutura MSG_RESULT contém o resultado de uma mensagem, isto é, se o destinatário existe ou não no jogo atualmente.
- A estrutura F_MSG_RESULT serve como “selo” para a estrutura MSG_RESULT armazenando uma flag e a estrutura MSG_RESULT
- A estrutura LABIRINTO armazena informações do jogo, tais como, o mapa, o número de blocos e de pedras no labirinto
- A estrutura JOGO que contém o estado atual do jogo, tendo assim a lista de jogadores, o nível do jogo, e a estrutura do LABIRINTO.
- A estrutura F_JOGO funciona como um “selo” para a estrutura de JOGO
- A estrutura TABELA armazena as informações dos jogadores atualmente disponíveis num array, para depois ser mostrado ao jogador em formato tabela
- A estrutura F_TABELA funciona como “selo” para o conteúdo da estrutura TABELA
- A estrutura ATUALIZA_JPLAYERS contem uma string com informações sobre a atualização de jogadores, usado para mostrar na secção de jogadores no *JogoUI* os jogadores atualmente no jogo
- A estrutura F_ATUALIZA_JPLAYERS serve como “selo” para o conteúdo da estrutura ATUALIZA_ JPLAYERS
- A estrutura NOTIFICACAO que contem uma string com algum tipo de informação para alertar os clientes da plataforma
- A estrutura F_NOTIFICACAO serve como “selo” para o conteúdo da estrutura NOTIFICACAO

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, assim como o uso de threads e selects, que facilitou a comunicação entre as várias tarefas. Em suma, esperamos ter conseguido cumprir com todos os requisitos para este trabalho.