Relatório Final

Trabalho Prático – Sistemas Operativos

Trabalho realizado por:
Diogo Oliveira - 2021146037
Lara Bizarro – 2021130066

Índice

Índice de Figuras

Introdução

No âmbito da cadeira de Sistemas Operativos (SO), foi pedido para realizar um jogo de Labirinto *Mutiplayer* com vários níveis. O objetivo dos jogadores é mover – se de um ponto inicial para o ponto final num labirinto em que poderão surgir obstáculos, a medida que o utilizador vai avançando, os mapas ficaram mais complexos.

Este trabalho aborda conhecimentos do sistema Unix, foi realizado na linguagem C para implementar na plataforma Unix (Linux).

O seguinte relatório está dividido em Estruturas de Dados e Constantes Simbólicas, implementação do Motor, implementação do JogoUI e conclusão.

Estruturas de Dados e Constantes Simbólicas

Abordamos este projeto, dividindo as Estruturas de Dados em três ficheiros, um ficheiro para o motor, neste existem as variáveis ambientes, que foram definidas em relação ao enunciado do Trabalho, contém todos os labirintos, um ponteiro para uma *string* com o número máximo de utilizadores definidos anteriormente. De seguida a inicialização das *threads* que iram ser usadas no Motor e ponteiros para as Pedras para as Barreiras e para as variáveis Ambiente.

```c
#ifndef Motor_H
#define Motor_H
#include <pthread.h>

#define MAXUSER 5
#define MAXBOT 10
#define MAXPEDRA 50
#define MAXNIVEL 3
#define MAXBMOVEL 5
#define MAXY 16
#define MAXX 40

#define NFICHEIRO1 "labirinto1.txt"
#define NFICHEIRO2 "labirinto2.txt"
#define NFICHEIRO3 "labirinto3.txt"

#define INSCRICAO "20"
#define NPLAYERS "1"
#define DURACAO "70"
#define DECREMENTO "20"

char *nomeUsers[MAXUSER] = {"","","","",""};
char letras[MAXUSER];
pid_t pidJogadores[MAXUSER];
int listaUsers[MAXUSER]={0};

int fdmotor;
int incricaoEncerrada = 0;
int segundosrestantes;

pthread_t TTempoInscricao;
pthread_t TMovel;
pthread_t TRecebeJogador;
pthread_t TComandos;
pthread_t TRecbeInfo;
pthread_t TBots;
pthread_t TEnviaInfo;
pthread_mutex_t mutexInfo = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutexTempoInscricao =  PTHREAD_MUTEX_INITIALIZER;

typedef struct pedra pedra,*ppedras;
typedef struct barreira barreira,pbarreira;
typedef struct va va, *pva;

struct va{
    int inscricao;
    int decremento;
    int nplayers;
    int duracao;
    char ficheiro[100];
};
```

Figura 1 - Estrutura do Motor

Ao nível da estrutura para o JogoUI, existem duas variáveis que ditam o comprimento e a largura da janela do jogo, consta também dois ponteiros para as janelas e para as mensagens, referente as estas duas *struct´s*. Contém a inicialização das threads para receber informação do Motor, enviar informação para o Motor, comunicar entre jogadores e receber informação do teclado e por fim as funções para validar os comandos, desenhar o labirinto, desenhar uma janela para os comandos, receber do teclado e receber mensagens.

```c
#ifndef JogoUI_H
#define JogoUI_H
#include <pthread.h>
#include <ncurses.h>

#define MAXY 16
#define MAXX 40

#define JOGOENVIARJ "jogoenviarjogos%d"

char JOGADORES_DESTINO[100];
typedef struct janelas janelas,*pjanelas;
typedef struct MSG MSG,*pmsg;

struct MSG{
    char mensagem[500];
    char user[100];
};
struct janelas{
    WINDOW *janelaTopo;
    WINDOW *janelaBaixo;
};
/* typedef thread{

}; */
pthread_t TRecbeMotor;
pthread_t TEnviaMotor;
pthread_t TComunicacaoJogadores;
pthread_t TTeclado;

//Função para a Validação do Comandos
void ValidacaoComandos();
//Função para desenhar o labirinto
void DesenhaLabirinto(WINDOW *janela, int tipo, char labirinto[MAXY][MAXX]);
//Função que mostra as mensagens do comando
void DesenhaComandos(WINDOW *janela, int tipo);
//Função que recebe teclas
void trataTeclado(WINDOW *janelaTopo, WINDOW *janelaBaixo);
//Função da thread que recebe mensagens do motor
void *RecebeMensagens(void* janelas);
//Função da thread que recebe inputs
void *RecebeTeclado(void* janelas);
#endif
```

Figura 2 - Estruturas do JogoUI

Por último existe uma estrutura partilhada, para não haver informação repetida, está contém uma *struct* que contém o processo das threads, e o utilizador e uma struct, que abrange variáveis para o Motor e para o JogoUI.

```c
#ifndef ESTRUTURAS_H
#define ESTRUTURAS_H
#include <stdbool.h>

#define MAXY 16
#define MAXX 40
#define MAXUSER 5

#define MOTOR "motorfifo"
#define JOGO "jogofifo%d"
#define LABIRINTO "labirintofifo%d"

char JOGO_DESTINO[100];
char LABIRINTO_DESTINO[100];
typedef struct jogo jogo, *pjogo;

typedef struct{
    pid_t pid;
    char user[100];
    int userValid;
}username;

struct jogo{
    //motor stuff
    int nivel;
    int pedras;
    int barreira;
    bool alteracao;
    pid_t pJogadores[MAXUSER];
    pthread_mutex_t mutex;
    char labirinto[MAXY][MAXX];
    char MSG_Motor[1000];
    char *Jogadores[MAXUSER];
    //jogador stuff
    pid_t pid;
    int movimento[3]; //manda array com 4 espaços 0000, 1000- cima, 0100- direita, 0010- baixo, 0001-esquerda
    bool ListaJogadores; //pede o players true ou false
};

#endif
```

Figura 3 - Estruturas

Implementação do Motor

Primeira são criadas as variáveis Ambiente, é verificado se existem e se não existem é dado um valor.

De seguida são criadas três, uma para receber o nome do utilizador, a outra envia a validação do nome do utilizador e recebe os comandos do motor.

É criado um alarme para o tempo de jogo.

Por fim são criadas as *threads*, uma que envia recebe que os utilizadores, outra que lança os bots. Cada *Tthead* tem associada uma função da mesma.

Implementação do JogoUi

Primeiramente é criado os *named pipes*, enviado o nome do utilizador para o motor, se for valido avança, senão morre o processo.

Existe a iniciazação do Ncurses, recebe o mapa do Motor e mostra ao utilizador, de seguida existe a possibilidade de movimentação do utilizador.

Conclusão

Na realização do jogo de Labirinto Multiplayer com vários níveis no âmbito da disciplina de Sistemas Operativos (SO), foram explorados e aplicados diversos conceitos fundamentais do sistema Unix. As várias estratégias usadas contribuíram para a realização do projeto.

Ao finalizar este projeto, é possível perceber a interligação entre os conceitos estudados em Sistemas Operativos e a prática da linguagem C no ambiente Unix foi essencial para o desenvolvimento do trabalho.

Em suma, a implementação do Jogo Labirinto Multiplayer não só desafiou a habilidade técnica e conceitos dos sistemas operativos Unix e da linguagem C, mas também proporcionou um contexto prático para a aplicação desses conhecimentos.