Licenciatura em Engenharia Informática

Unidade Curricular Sistemas Operativos

Ano letivo 23/24

Programação em C para Unix

David José Nobre Pires – a2019129618@isec.pt

Dinis Meireles de Sousa Falcão – a2020130403@isec.pt

Índice

Introdução

Este trabalho prático consiste em programar em C para Unix, um jogo de labirinto multijogador com vários níveis. Sendo o objetivo o jogador deslocar se do ponto de partida até um ponto final evitando obstáculos que surgem em posições aleatórias.

O trabalho foca, acima de tudo, gestão de processos, comunicação entre processos e a execução de tarefas em paralelo.

Relativamente à meta 1, foram feitas várias alterações ao código de modo a concluir o trabalho para esta meta.

Estruturas Usadas

Considerámos o uso de várias estruturas para a implementação do nosso trabalho prático, uma estrutura Jogador que guarda informações relevantes sobre o jogador.

Estrutura para armazenar todas as estruturas Jogador num array, denominada por ArrayJogadores. Uma estrutura Bot, que guarda informação sobre o bot e posteriormente uma estrutura ArrayBots que tem a funcionalidade de guardar as estruturas Bot num array de estruturas.

À semelhança da estrutura Bot e ArrayBots, temos também uma estrutura Bmov e uma estrutura Pedra que guardam respetivamente informação sobre as posições e durações. Existe igualmente uma estrutura ArrayBmov que tem como objetivo armazenar num array estruturas do tipo Bmov.

Para guardar informação relativa ao mapa, possuímos uma estrutura Mapa, que guarda o mapa em si e também um array de estruturas do tipo Pedra.

Temos também uma estrutura responsável ir juntado informação proveniente do teclado denominada Keyboard. Por fim temos uma estrutura apenas para ajudar na sincronização, relativamente aos inícios dos jogos.

Relativamente quanto ao nível do jogo em questão temos uma estrutura JogadorNivel que auxilia nesse aspeto.

Uma union para guardar diferentes tipos de dados no mesmo sítio, e uma enumeração Messagem. Por fim duas estruturas: MensagemJogadores e JuntaInfoBot, respetivamente a primeira auxilia nas mensagens dos jogadores enquanto a segunda estrutura ajuda na 'recolha' de informação proveniente dos Bots.

Estratégia de Implementação

Para resolver os vários e diferentes problemas propostos pelo trabalho prático decidimos desenvolver diferentes estratégias.

Em relação à questão da comunicação entre o motor e o bot, esta foi resolvida através da criação de um unnamed pipe em que mandamos executar o programa bot e recebemos o output guardando-o.

As restantes comunicações foram feitas à base de fifos associados ao pid.

Para a gestão dos file descriptors desenvolvemos o mecanismo select do lado do motor enquanto que do lado do jogoUI foi utilizada thread.

Esta thread no jogoUI possibilita nos a receção de informação que vem do motor e também a gestão do tratamento do teclado através dos comados escritos.

No motor.c podemos também encontrar algumas threads que facilitam a nossa tarefa. Tais como, thread para tratar do input de comandos pelo teclado; thread para gerir os bots em cada nível; uma thread para gerir a colocação no mapa de bots e outras duas threads com o mesmo objetivo, mas para os bloqueios móveis e pedras respetivamente. E uma última thread que gere os eventos de jogo.

De referir que a biblioteca ncurses apenas foi utilizada para interface do utilizador, encontrando se implementada no jogoUI.c.

Makefile

```
all: motor jogoUI bot

motor: motor.o funcoes.o
    gcc motor.o funcoes.o cursesHelpers.o -pthread -lncurses -o motor

jogoUI: jogoUI.o funcoes.o
    gcc jogoUI.o funcoes.o cursesHelpers.o -pthread -lncurses -o jogoUI

bot: bot.o
    gcc bot.o -o bot

motor.o: motor.c funcoes.h structs.h
    gcc -c motor.c

jogoUI.o: jogoUI.c funcoes.h structs.h
    gcc -c jogoUI.c

bot.o: bot.c
    gcc -c bot.c

funcoes.o: funcoes.c funcoes.h commons.h
    gcc -c funcoes.c

cursesHelpers.o: cursesHelpers.c cursesHelpers.h commons.h
    gcc -c cursesHelpers.c

clean:
    rm *.o motor jogoUI bot
```

Conclusão

Tendo em vista a conclusão deste trabalho prático, adquirimos vários conhecimentos sobre como funciona melhor a comunicação entre processos. E como este trabalho prático permitiu-nos colocar em prática conceitos importantes sobre a arquitetura, criação e desenvolvimento de programas para sistemas UNIX.