Projeto de Sistemas Operativos 2025-26
Enunciado da 1ª parte
LEIC-A/LEIC-T/LETI

O objetivo deste projeto é desenvolver o PacmanIST, um jogo bi-dimensional em que uma personagem percorre um labirinto recolhendo pontos e fugindo de monstros. O jogo será jogado sobre uma matriz rectangular em que existirão posições que serão espaços livres para circulação e espaços que serão paredes. Cada espaço livre pode estar ocupado por um ponto, um portal, um monstro, um Pacman ou pode estar vazio. Pelos espaços livres circularão personagens que serão ou Pacmen ou monstros usando movimentos ortogonais (cima/baixo/direita/esquerda). Se um Pacman passar por um espaço com um ponto esse ponto é recolhido tornando esse espaço num espaço vazio. Se um Pacman se mover num espaço ocupado por um monstro, o Pacman morre e o jogo termina. O estado do jogo estará armazenado na memória do programa que recebe os comandos descritos abaixo e armazena os dados.

O PacmanIST explora técnicas de paralelização baseadas em múltiplas tarefas de forma a acelerar a execução do jogo e recorre a processos para fazer cópias de segurança dos dados de forma não bloqueante. Ao desenvolver o PacmanIST, os estudantes aprenderão também como implementar mecanismos de sincronização escaláveis entre tarefas bem como mecanismos de comunicação entre processos (FIFOs e sinais). O PacmanIST irá também interagir com o sistema de ficheiros oferecendo portanto a possibilidade de aprender a utilizar as interfaces de programação de sistemas de ficheiros POSIX. Na segunda parte do projecto, haverá um programa servidor do jogo que permitirá a clientes ligarem-se remotamente ao jogo.

Código base

O código base fornecido disponibiliza uma implementação sequencial que aceita os seguintes comandos para movimentar a personagem do utilizador, o Pacman: as teclas W (cima), A (baixo), S (esquerda) e D (direita).
O Pacman é representado no tabuleiro de jogo pelo carácter “C”, enquanto os monstros são representados pelo “M”. No código base o comportamento dos monstros está definido de forma estática (algo que será alterado no exercício 1). Os comportamentos previstos para os monstro são movimentos ortogonais (tal como pelo Pacman) e uma ação adicional “Carregar”. Quando um monstro usa a ação

“Carregar”, o próximo movimento será feito em linha ao invés de somente uma célula.. No código base o tabuleiro inclui também um portal, representado pelo carácter “@” que faz com que o Pacman salte automaticamente para o próximo nível (que no código base coincide com uma nova instância do mesmo nível, algo que será alterado também no exercício 1)

1ª parte do projeto

A primeira parte do projeto consiste em 3 exercícios.

Exercício 1. Interação com o sistema de ficheiros

O código base aguarda comandos para movimentar o Pacman enquanto que os monstros se movem aleatoriamente. Em termos do cenário (paredes e espaços livres) o código base usa um único cenário que se repete indefinidamente (voltando ao estado inicial depois do Pacman atravessar o portal). Neste exercício pretende-se estender o código base de forma que passe a processar dados obtidos a partir de ficheiros, quer o comportamento dos monstros e do Pacman quer o desenho dos níveis do jogo.
Para este efeito o PacmanIST deve passar a receber como argumento na linha de comando o caminho para uma diretoria, onde se encontram armazenados os ficheiros de entrada. O PacmanIST deverá obter a lista de ficheiros com extensão .m, .p e .lvl contidos na diretoria indicada no primeiro parâmetro. Os ficheiros .lvl contêm uma descrição de cada um dos níveis do jogo (ver sintaxe abaixo). Os ficheiros .m contêm o comportamento dos monstros de cada nível. Os ficheiros .p, se existirem, conterão a descrição do comportamento do Pacman. Se não existirem, assume-se que o Pacman será movido manualmente pelo utilizador.
Com esta extensão do código base, o jogo termina quando o Pacman atravessar o portal do último nível disponível.

O acesso e a manipulação de ficheiros deverão ser efetuados através da interface POSIX baseada em descritores de ficheiros, e NÃO usando a biblioteca stdio (carregada através do ficheiro stdio.h) e a abstração de *FILE stream.*

Sintaxe dos ficheiros de entrada

Todas as linhas iniciadas por # são comentários e devem ser ignoradas. Por simplicidade, assume-se que não há erros de sintaxe ou inconsistências nos ficheiros de entrada.

Exemplo de ficheiro de nível, e.g. "1.lvl"*:*

```
# DIM: dimensões da matriz do nível
# só existe um no início por ficheiro
DIM 6 6
# TEMPO: indica a duração de cada jogada em milissegundos
TEMPO 10
# PAC: (opcional) indica o nome do ficheiro do pacman deste nível
PAC 1.p
# MON: indica os nomes dos ficheiros de monstros usados neste nível
MON 1.m 3.m
# o ficheiro termina com o conteúdo da matriz de jogo em que "X"
# representa uma parede, "o" representa um espaço de circulação
# e “@” representa o portal
XXoXXX
XoooXX
XoXoXX
Xooo@X
XooooX
XXXXXX
```

Exemplo de ficheiro de comportamento de um Pacman, e.g. "1.p" ou de um monstro , e.g "2.m"*:*

```
# PASSO: comando de espaçamento de movimentos
# Só existe um no início por ficheiro. Indica quantas jogadas esperar
# entre cada movimento.
PASSO 1
# POS: comando de colocação inicial do monstro (linha e coluna).
# Assume-se que não é possível o monstro ser colocado
# numa posição impossível/inexistente.
# Só existe um no início por ficheiro.
POS 4 1
# Todos os comandos após PASSO e POS são executados em ciclo infinito.
# Os comandos possíveis são A (esq.), D (dir.), W (cima.), S (baixo)
# R (direcção aleatória), T (espera um número de jogadas), C (carregar)
A
A
D
```

```
T 2
W
S
S
S
```

Exercício 2. Reencarnação do Pacman

Após terem realizado o Exercício 1, os alunos devem estender o código criado de forma a que sempre que um jogador prima a tecla "G", o sistema guarde o estado atual do jogo. Assim, caso o Pacman seja apanhado por um monstro, o jogo será reiniciado a partir do estado guardado anteriormente.

Este procedimento deve ser implementado usando a chamada de sistema fork() para que seja possível explorar o sistema operativo para guardar o estado do jogo (no processo pai ou no processo filho?) e para retomar a execução do jogo a partir deste estado após a morte do Pacman.

Só pode existir 1 estado guardado a qualquer ponto do jogo. Ou seja, se o jogador premir a tecla “G” quando já existe um estado guardado, nada deve acontecer.

Exercício 3. Paralelização usando múltiplas tarefas

Neste exercício pretende-se permitir que os monstros possam executar sem terem de esperar pelo input do pacman. Para isso, cada personagem deve ser gerida por uma tarefa (*thread*) separada.

AVISO: A biblioteca <ncurses.h> **NÃO** é *thread-safe*. Visto isto, as *threads* que controlam os personagens não podem todas interagir com as funções do ncurses. Pretende-se portanto que exista apenas uma *thread* responsável por interagir com a biblioteca ncurses para alterar a visualização do tabuleiro, enquanto as outras *threads* são responsáveis por atualizar o estado do tabuleiro.

Serão valorizadas soluções de sincronização no acesso ao estado do jogo que maximizem o grau de paralelismo atingível pelo sistema.

Idealmente, este exercício deveria ser realizado a partir do código obtido após a resolução do exercício 2. Contudo, não serão aplicadas penalizações se a solução deste exercício for realizada a partir da solução do exercício 1.

Submissão e avaliação

A submissão é feita através do Fénix **até ao dia 12/12/2025 às 23h59**.

Os alunos devem submeter um ficheiro no formato zip com o código fonte e o ficheiro Makefile. O arquivo submetido não deve incluir outros ficheiros (tais como binários). Além disso, o comando make clean deve limpar todos os ficheiros resultantes da compilação do projeto.

Recomendamos que os alunos se assegurem que o projeto compila/corre corretamente no cluster *sigma*. Ao avaliar os projetos submetidos, em caso de dúvida sobre o funcionamento do código submetido, os docentes usarão o cluster *sigma* para fazer a validação final.

O uso de outros ambientes para o desenvolvimento/teste do projeto (e.g., macOS, Windows/WSL) é permitido, mas o corpo docente não dará apoio técnico a dúvidas relacionadas especificamente com esses ambientes.

A avaliação será feita de acordo com o método de avaliação descrito no *site* da cadeira.

Os alunos não podem partilhar código e ou soluções com outros grupos. O código submetido tem de ser o resultado do trabalho original de cada grupo. A submissão de código com grande grau de semelhança com outros grupos ou realizado recorrendo a entidades externas ao grupo levará à reprovação dos grupos envolvidos e ao reporte da situação à coordenação da LEIC e ao Conselho Pedagógico do IST.