UNIVERSIDADE da MADEIRA

Faculdade de Ciências Exatas e da Engenharia

Sistemas Operativos

Projeto Prático, 2017/2018, 1º Semestre

Simulação de uma Montanha-Russa

Docentes
Eduardo Marques | Luís Gaspar
(emarques@uma.pt | lgaspar@staff.uma.pt)

1 Objetivos

O presente trabalho tem como objetivo a concepção e a implementação de um simulador de uma Montanha-Russa . Pretende-se assim que os alunos pratiquem os conceitos aprendidos nas aulas teóricas e práticas na concepção e desenvolvimento de um sistema simples, empregando os mecanismos de concorrência, sincronização e comunicação existentes na linguagem C.

2 Descrição

Uma forma de divertimento das pessoas é a realização de viagens em Montanhas-Russas. Junto à Montanha-Russa deverá existir um local de venda de bilhetes e, junto à entrada, alguém verificará que apenas se dirigem aos carros um número máximo de pessoas (capacidade do(s) carro(s)). Por vezes as pessoas cansam-se de esperar e/ou têm receio de fazer a viagem e podem desistir.

Após cada viagem todos devem sair dos carros e, na zona dos carros, já devem estar os próximos passageiros.

A simulação a implementar pretende avaliar as condições de funcionamento de um equipamento deste género em termos de quantas pessoas pode servir, quais os tempos de espera médios, quantas desistem, o número de carros ótimo, entre outros.

As opções para a simulação são muitas e variadas e ficam à consideração de cada grupo, constituindo elemento de avaliação. Por exemplo: Qual o tamanho máximo da fila para o guiché? Existe alguma prioridade (carros da frente e carros traseiros)? As pessoas chegam todas de uma vez ou vão chegando? O tempo de viagem é sempre o mesmo? O tempo de saída/entrada nos carros é relevante?

3 Arquitetura

O sistema a desenvolver deverá conter duas aplicações, a primeira (Simulador) que efetuará toda a simulação, e a segunda (Monitor) que receberá todas as mensagens enviadas pela primeira e fará todo o seu tratamento.

O **Simulador** deverá ser lançado tendo por parâmetro o ficheiro de configuração da simulação. Os dados que deverão estar presentes, no mínimo, para o início da simulação são os seguintes: tempo médio[1] de chegada dos utilizadores e tempos diversos, dimensão dos recursos, probabilidade de desistência nas filas, início da simulação e tempo de simulação[2]. A Figura 1 apresenta a arquitetura de ficheiros para o projeto. Caso necessário podem ser indicados outros ficheiros (um ficheiro por cada tipo de relatório, apenas um ficheiro de configuração, e outros).

O **Simulador** deverá no seu arranque ligar-se ao **Monitor**. Durante a simulação devem ser apresentados alguns dados sobre o estado da simulação. Por exemplo:

```
  ...
Chegou um utilizador Numero 234.
O utilizador 123 comprou um bilhete.
O utilizador 136 entrou no carro.
O utilizador 129 desistiu.
  ...
```

O **Simulador** deverá ter métodos para gerar aleatoriamente a chegada dos utilizadores (cada utilizador será um *thread*) aos recursos, que permitam a correta coordenação e sincronização dos utilizadores (por via de semáforos) e que enviem mensagens para o **Monitor** (comunicação via *sockets* – Unix ou Internet).

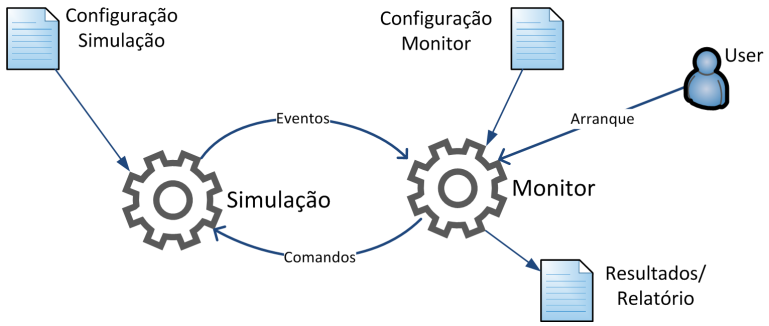

Figura 1: Arquitetura de ficheiros para o projeto

O **Monitor** aceita uma ligação de um Simulador e guarda em ficheiros todos os registos enviados por este. O tratamento dos dados da simulação acontece ao nível do Monitor (cálculo da média do tempo à espera, por exemplo). Durante o funcionamento do **Monitor** este deve apresentar dados sobre o seu estado de execução. Por exemplo:

```
Estado actual => Simulacao a decorrer!
Utilizadores: 203
Desistencias: 18
N.o de Viagens: 11
Tempo medio de espera: 13 min.
```

[1] Os tempos médios expressam o ritmo aproximado de um dado acontecimento e deve ser implementado um método aleatório para calcular os diversos tempos para diversas fases do projeto.

[2] A temporização da simulação deve ser definida de forma a poder ser testada em poucos minutos. Aconselha-se a utilização do *time stamp* do unix e resumir os tempos da simulação a segundos.

Exemplo dos comandos para lançamento das aplicações:

```
$ monitor monitor.conf
$ simulador simulacao.conf
```

4 Estrutura de Dados

As duas aplicações terão um conjunto de estruturas de dados para guardar diversas informações. Para os ficheiros de configuração fica a seguinte sugestão:

```
PARAMETRO1:dado1
PARAMETRO2:dado2
...
```

onde 'PARAMETRO' significa a designação do parâmetro e 'dado' é o seu valor.

Os dados sobre os acontecimentos da simulação poderão ter o seguinte formato:

```
IdUtilizador hora Acontecimento
...
```

onde o 'Acontecimento'[3] pode ser, por exemplo, um dos seguintes: criação do utilizador, chegada ao recursos, passagem em filas, entrada e saída dos recursos e desistência.

5 Entrega e Avaliação do Projeto

O projeto será desenvolvido em grupo e a sua constituição será divulgada nas aulas e na página da disciplina. O projeto será dividido em três fases (obrigatórias) e as datas limite de cada fase serão anunciadas nas aulas e/ou na página da cadeira.

5.1 1ª Fase

A primeira fase deverá conter as bibliotecas para a gestão da informação nos ficheiros de texto. As aplicações **Monitor** e **Simulador** já devem carregar os parâmetros para a sua configuração e ainda exportar para um (ou vários) ficheiro(s) o registo dos eventos que estão a ocorrer na simulação. É aconselhável já estar implementada nesta fase a criação de tarefas no simulador e monitor.

A avaliação recairá sobre o código implementado e as estruturas de dados escolhidas para a configuração e para os registos em ficheiros. Na aula seguinte à data definida para esta fase os grupos deverão demonstrar o funcionamento do código. A avaliação desta fase tem um peso de **10%** na nota final da parte prática.

5.2 2ª Fase

A segunda fase envolve a implementação das bibliotecas para a comunicação entre o **Simulador** e o **Monitor**; e, o interface com o utilizador, onde será apresentado o estado corrente da simulação. Deverá ainda ser entregue um relatório que descreva e fundamente (de forma simples) as opções seguidas, principalmente no formato das mensagens, onde deve ser apresentado o protocolo de comunicação entre o cliente e o servidor.

[3] Sugestão: codificar cada acontecimento como um inteiro.

O relatório deverá ainda conter uma descrição das funcionalidade a implementar, bem como a forma como se pretende resolver a questão da sincronização. O relatório deverá ser conciso e explicar brevemente cada funcionalidade. A entrega desta fase será feita na plataforma Moodle, na área da disciplina.

A avaliação desta fase recairá sobre o código implementado, nomeadamente as estruturas do protocolo de comunicação e o *interface* com o utilizador já definido. Será ainda parte da avaliação o nível de maturidade da solução proposta para a sincronização da simulação. Na aula seguinte à data definida para esta fase os grupos deverão demonstrar o funcionamento do código e descrever rapidamente a sua visão do problema. A avaliação desta fase tem um peso de **20%** na nota final da parte prática.

5.3 3ª Fase

Por fim, a terceira fase, deverá agregar todas as bibliotecas anteriores e ainda conter o código com os mecanismos e políticas de sincronização. Nesta fase deverá ser ainda entregue a fundamentação da solução escolhida e as conclusões gerais do trabalho sob a forma de um relatório completo do projeto. A avaliação desta fase tem um peso de **70%** na nota final da parte prática, divididos (não igualmente) entre o código desenvolvido, a apresentação/defesa do projeto e o relatório.

O relatório deve ainda incluir uma resposta/reflexão às seguintes questões:

- Qual a influência de uma alteração do padrão de chegadas na solução apresentada?
- A solução apresentada apresenta um maior preocupação no uso justo/equilibrado dos recursos ou na eficiência geral do sistema?
- Descrevam, pelo menos, duas limitações da solução apresentada.

O relatório deve ser entregue via Moodle (que deve também conter em anexo a listagem do código fonte, mas formatado de forma a reduzir ao máximo o número de folhas sem perder legibilidade). Os relatórios devem ainda ser entregues em papel e com a indicação clara na capa do(s) nome(s) do(s) aluno(s), número(s) mecanográfico(s) e curso(s), além dos dados do projeto.

As apresentações/discussões serão na semana após a da entrega do trabalho. O código e o relatório serão lidos pelo docente, e defendidos pelos alunos numa apresentação/discussão do projeto.

A apresentação do trabalho é por grupo, mas a defesa e a nota final é individual, pelo que a não comparência de um elemento em qualquer uma fases das implica nota "zero"no projeto.

Terão de entregar junto com o código fonte uma *Makefile* que compile corretamente as aplicações desenvolvidas. Terão de entregar um ficheiro compactado (ZIP ou RAR) contendo todo o código necessário para compilar e executar o programa. A este ficheiro deve ser acrescentado um ficheiro *README.TXT* de ajuda à compilação e execução do sistema para que um utilizador possa ler, compilar e visualizar o trabalho sem a sua ajuda dos seus autores.

6 Considerações Finais

A implementação do sistema poderá ser executada utilizando qualquer ambiente de desenvolvimento da linguagem C disponível. Recomenda-se no entanto a utilização do ambiente usado nos laboratórios.

Algumas considerações gerais:

- É preferível apresentarem um projeto que funciona mas que não cumpre com todas as funcionalidades básicas do que apresentar um projeto que supostamente implementa tudo mas não funciona;
- O relatório serve para descrever o que foi feito e, principalmente, fundamentar as opções tomadas. O relatório deve ainda conter os testes às aplicações e dados sobre as simulações executadas, não deixando de ser feita uma análise aos resultados obtidos;
- Evitem relatórios extensos, com erros ortográficos, com capas coloridas e/ou outros adereços complemente desnecessários, repetir partes do enunciado no relatório, etc. Lembrem-se que o que conta não é o número de páginas mas sim a qualidade do conteúdo.