

Projeto *Scut-IUL* (Parte 1)

O presente trabalho visa aplicar os conhecimentos adquiridos durante as aulas de Sistemas Operativos e será composto por três partes, com o objetivo de desenvolver os diferentes aspetos da plataforma **Scut-IUL**. Iremos procurar minimizar as interdependências entre partes do trabalho.



Este enunciado detalha apenas as funcionalidades que devem ser implementadas na parte 1 do trabalho.

A plataforma **Scut-IUL** destina-se à gestão da utilização de um sistema de pagamento automático de portagens. Na Plataforma **Scut-IUL**, existem os seguintes conceitos:

- **Veículo:** veículo conduzido por um condutor, registado na plataforma, caracterizado pelos seguintes dados: Marca, Modelo, Cor, Matrícula e ID do condutor associado.
- **Condutor:** a pessoa que conduz o veículo. Um condutor tem de estar registado na plataforma definida **Scut-IUL** Com os seguintes dados – ID, Nome, Nº Carta, Contacto, Contribuinte, Saldo (em créditos).
- **Portagem:** local de cobrança da taxa de utilização da autoestrada. A portagem é caracterizada por: ID de Portagem, Lanço, Autoestrada atribuída, Taxa de utilização (em créditos).
- **Relatório de utilização:** relatório emitido, contendo a descrição de cada utilização das portagens registadas no sistema, contendo o ID de Portagem, Lanço, ID do condutor associado, Matrícula do veículo, Taxa cobrada, e Data da utilização.

Em termos gerais, um novo condutor regista-se na plataforma **Scut-IUL** fornecendo os seus dados, bem como os do veículo que possui. Ao efetuar o registo, é atribuído a cada condutor um valor de 150 créditos. Após o registo, o condutor poderá passar a utilizar o sistema de cobrança automático de portagens. Sempre que o condutor passar por uma portagem, é descontado o valor da taxa de utilização ao respetivo saldo e adicionada uma linha no relatório de utilização. As portagens são previamente registadas na plataforma, contendo a informação descrita acima.

Os alunos não deverão usar o programa **echo** (não será analisado para efeito de avaliação) para os outputs no ecrã, mas sim os scripts **success** (para as mensagens de sucesso) e **error** (para as mensagens de erro), devendo analisar os respetivos scripts para ver exemplos de invocação dos mesmos.

Procedimentos de entrega e submissão do trabalho

O trabalho de SO será realizado **individualmente**, logo sem recurso a grupos.

A entrega da Parte 1 do trabalho será realizada através da criação de **um** ficheiro ZIP cujo nome é o nº do aluno, e.g., “a<nºaluno>-parte-1.zip” (**ATENÇÃO: não serão aceites ficheiros RAR, 7Z ou outro formato**) onde estarão todos os ficheiros criados. Estes serão **apenas** os ficheiros de código, ou seja, na primeira parte, apenas os ficheiros de código (*.sh). Cada um dos módulos será desenvolvido com base nos ficheiros fornecidos, e que estão na diretoria do Tigre “/home/so/trabalho-2021-2022/parte-1”, e deverá incluir nos comentários iniciais um “relatório” indicando a descrição do módulo e explicação do mesmo (poderá ser muito reduzida se o código tiver comentários bem descritivos). Naturalmente, deverão copiar todos estes ficheiros para a vossa área, e não editar os ficheiros da diretoria /home/so/trabalho-2021-2022.

Para criarem o ficheiro ZIP, usem, no Tigre, o comando **\$ zip a<nº aluno>-parte-1.zip <ficheiros>**, por exemplo:
\$ zip a123456-parte-1.zip *.sh

O ficheiro ZIP deverá depois ser transferido do Tigre para a vossa área local (Windows/Linux/Mac) via SFTP.

Antes de submeter, por favor validem que o ficheiro ZIP não inclui diretorias ou ficheiros extra indesejados.

A entrega desta parte do trabalho deverá ser feita por via eletrónica, através do e-learning:

- e-learning da UC Sistemas Operativos, Seleccionam a opção sub-menu “Conteúdo/Content”;
- Seleccionem o link “Trabalho Prático 2021/2022 Parte 1”;
- Dentro do formulário “Visualizar Exercício de carregamento: Trabalho Prático 2021/2022 Parte 1”, seleccionem “Anexar Arquivo” e anexem o vosso ficheiro .zip. Podem submeter o vosso trabalho as vezes que desejarem. **Apenas a última submissão será contabilizada.** Certifiquem-se que a submissão foi concluída, e que esta última versão tem todas as alterações que desejam entregar dado que os docentes apenas considerarão esta última submissão;
- Avisamos que a hora *deadline* acontece sempre poucos **minutos antes da meia-noite**, pelo que se urge a que os alunos não esperem por essa hora final para entregar e o façam antes, idealmente um dia antes, ou no pior dos casos, pelo menos uma hora antes. **Não serão consideradas válidas as entregas realizadas por e-mail.** Poderão testar a entrega nos dias anteriores para perceberem se têm algum problema com a entrega, sendo que, reiterando, **apenas a última submissão conta.**

Política em caso de fraude

O trabalho entregue deve corresponder ao esforço individual de cada aluno. São consideradas fraudes as seguintes situações: Trabalho parcialmente copiado, facilitar a cópia através da partilha de ficheiros, ou utilizar material alheio sem referir a sua fonte.

Em caso de deteção de algum tipo de fraude, os trabalhos em questão não serão avaliados, sendo enviados à Comissão Pedagógica da escola (ISTA) ou ao Conselho Pedagógico do ISCTE, consoante a gravidade da situação, que decidirão a sanção a aplicar aos alunos envolvidos. Serão utilizadas as ferramentas *Moss* e *SafeAssign* para deteção automática de cópias. Recorda-se ainda que o Anexo I do Código de Conduta Académica, publicado a 25 de janeiro de 2016 em Diário da República, 2ª Série, nº 16, indica no seu ponto 2 que quando um trabalho ou outro elemento de avaliação apresentar um nível de coincidência elevado com outros trabalhos (percentagem de coincidência com outras fontes reportada no relatório que o referido software produz), cabe ao docente da UC, orientador ou a qualquer elemento do júri, após a análise qualitativa desse relatório, e em caso de se confirmar a suspeita de plágio, desencadear o respetivo procedimento disciplinar, de acordo com o Regulamento Disciplinar de Discentes do ISCTE - Instituto Universitário de Lisboa, aprovado pela deliberação nº 2246/2010, de 6 de dezembro.

O ponto 2.1 desse mesmo anexo indica ainda que no âmbito do Regulamento Disciplinar de Discentes do ISCTE-IUL, são definidas as sanções disciplinares aplicáveis e os seus efeitos, podendo estas variar entre a advertência e a interdição da frequência de atividades escolares no ISCTE-IUL até cinco anos.

Parte I – Shell Script (bash)

Data de entrega: 20 março de 2022

Nesta fase do trabalho, o objetivo é criar um conjunto de scripts para administração e gestão do sistema:

Copie para a sua diretoria local todos os ficheiros que se encontram no servidor Tigre, especificamente na diretoria `/home/so/trabalho-2021-2022/parte-1`.

Atenção: Apesar do vários ficheiros necessários para a realização do trabalho serem fornecidos na diretoria do Tigre `"/home/so/trabalho-2021-2022/parte-1"`, assume-se que, para a sua execução, os scripts e todos os ficheiros de input e de output estarão todos **sempre** presentes na mesma diretoria, que não deve estar *hard-coded*, ou seja, os Shell scripts entregues devem correr em qualquer diretoria.

1) `lista_condutores.sh`

Este script deverá ler o ficheiro local `peessoas.txt`. Se o ficheiro não existir, dá **error 1** e termina. Caso contrário, assuma que este ficheiro respeita a seguinte sintaxe e exemplo:

`<ID carta condução>:<Nome>:<Nr Contribuinte>:<Contacto>`

`peessoas.txt`

```
L1234567:Manuel Silva:234580880:961112223
L1844374:Maria Abreu:215654377:916566777
```

O script então deve criar de novo o ficheiro `condutores.txt`, que respeita a seguinte sintaxe:

`<ID>-<Nome>;<ID carta condução>;<Contacto>;<Nr Contribuinte>;<Saldo (em créditos)>`

- O campo `<ID>` é criado a partir do `<Nr Contribuinte>` de cada pessoa existente no ficheiro `peessoas.txt`, colocando-lhe o prefixo "ID". Não é necessário fazer a validação deste campo;
- Os campos `<Nome>`, `<ID carta condução>`, `<Contacto>` e `<Nr Contribuinte>` são extraídos diretamente do ficheiro `peessoas.txt`. Não é necessário fazer qualquer validação destes campos;
- O campo `<saldo>` é sempre registado com o valor inicial de 150 créditos.

Exemplo de um ficheiro `condutores.txt` baseado no `peessoas.txt`, já com todos os campos preenchidos:

`condutores.txt`

```
ID234580880-Manuel Silva;L1234567;961112223;234580880;150
ID215654377-Maria Abreu;L1844374;916566777;215654377;150
```

No final da sua execução, o script deve mostrar toda a lista de condutores registados (**success 2**).

2) altera_taxa_portagem.sh

Este script deverá ler o ficheiro local **portagens.txt**, e é executado quando se pretende alterar a taxa de utilização cobrada numa determinada portagem. O script é executado com os seguintes argumentos: **<Lanço: string>**, **<Auto-Estrada: string>**, **<Novo_Valor_Taxa: número inteiro positivo>**.

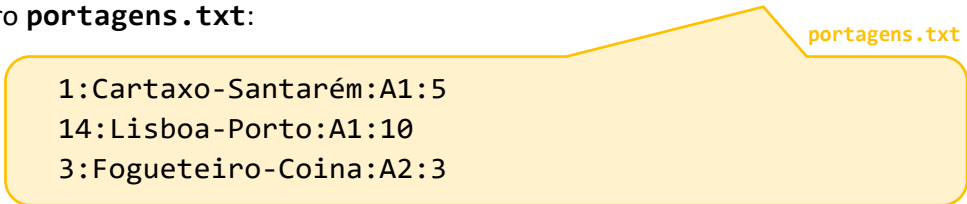
Exemplos de invocação deste script, que deverá receber os valores pedidos passados como argumentos:

```
$ ./altera_taxa_portagem.sh "Cartaxo-Santarém" "A1" 2
$ ./altera_taxa_portagem.sh "Leiria-Pombal" "A1" 2
$ ./altera_taxa_portagem.sh "Fogueteiro-Coína" "A2" 4
```

Com os campos inseridos como descrito acima, o script deve:

- Validar os argumentos passados, avaliando se são em número suficiente (se não, reportando o **error 2**) e se o formato do **<Lanço>** e do **<Novo_Valor_Taxa>** são os esperados (se não, reportando o **error 3**). Se houver erro, termina; Caso contrário, avança.
- Se o ficheiro **portagens.txt** existir, verifica se já existe o **Lanço** dessa **Portagem** registado na plataforma (no ficheiro **portagens.txt**);
- Se **não** existir esse **Lanço**, deverá utilizar os dados passados nos argumentos para registar um novo **Lanço** de portagem no ficheiro **portagens.txt**, seguindo a sintaxe:
<ID_portagem>:<Lanço>:<Auto-estrada atribuída>:<Taxa de utilização (em créditos)>
O campo **<ID_portagem>** é calculado a partir do maior **<ID_portagem>** registado, acrescido de 1 unidade. Se o ficheiro **portagens.txt** não existir, cria-o, e se estiver vazio, o primeiro **<ID_portagem>** é 1.
- Se já existir esse **Lanço** no ficheiro **portagens.txt**, independentemente do valor do argumento **Autoestrada**, deve apenas alterar a **Taxa de utilização** desse **Lanço** no ficheiro **portagens.txt**
- Mostrar uma mensagem (**success 3**) informando que a taxa de utilização do lanço foi atualizada.

Exemplo do ficheiro **portagens.txt**:



```
1:Cartaxo-Santarém:A1:5
14:Lisboa-Porto:A1:10
3:Fogueteiro-Coína:A2:3
```

Não deve ser possível executar este script sem passar todos os argumentos pedidos.

No final da sua execução, o script deve mostrar toda a lista de portagens (usando o programa fornecido **success 4**). Esta lista deverá estar ordenada primeiro por ordem alfabética (não é por ordem numérica!) de nome da **Autoestrada** e depois (dentro dos registos da mesma autoestrada), por ordem alfabética do nome do **Lanço**.

3) faturacao.sh

Este script será responsável por criar as faturas de cada condutor. Para isso, quando for executado, deve:

- Ler o ficheiro **relatorio_utilizacao.txt**. Se este ficheiro não existir, dá **error 1** e termina;
- Verificar se já existe o ficheiro **faturas.txt** e, caso exista, deve apagá-lo.
- No caso de haver registos no ficheiro **relatorio_utilizacao.txt**, deve ser criado o ficheiro **faturas.txt** onde serão agrupadas todas as utilizações de portagens de cada condutor e será apresentado o valor total cobrado a esse condutor.

Exemplo:

Considere, como exemplo, o seguinte ficheiro **relatorio_utilizacao.txt** com o formato:
<ID Portagem>:<Lanço>:<ID Condutor>:<Matrícula>:<Taxa_Portagem>:<Data>

relatorio_utilizacao.txt

```
1:Cartaxo-Santarém:ID234580880:12-HT-62:5:21/02/2022
14:Lisboa-Porto:ID234580880:12-HT-62:10:17/02/2022
1:Cartaxo-Santarém:ID215654377:00-MV-88:5:15/02/2022
1:Cartaxo-Santarém:ID215654377:00-MV-88:5:13/02/2022
14:Lisboa-Porto:ID234580880:12-HT-62:10:05/02/2022
```

Neste caso, devem ser criadas as faturas de cada um dos condutores que utilizou o sistema de portagens, listando os cliente pela mesma ordem que os mesmos estão no ficheiro **pessoas.txt**, e as faturas respetivas de cada cliente listadas pela mesma ordem que estão no ficheiro **relatorio_utilizacao.txt**.

Assim, o ficheiro **faturas.txt** ficará com o seguinte conteúdo:

faturas.txt

```
Cliente: Manuel Silva
1:Cartaxo-Santarém:ID234580880:12-HT-62:5:21/02/2022
14:Lisboa-Porto:ID234580880:12-HT-62:10:17/02/2022
14:Lisboa-Porto:ID234580880:12-HT-62:10:05/02/2022
Total: 25 créditos

Cliente: Maria Abreu
1:Cartaxo-Santarém:ID215654377:00-MV-88:5:15/02/2022
1:Cartaxo-Santarém:ID215654377:00-MV-88:5:13/02/2022
Total: 10 créditos
```

No final, o script deverá apresentar uma listagem de todas as faturas (**success 5**).

4) stats.sh

Este script é executado para obter informações sobre o sistema.

Nomeadamente, deve devolver a seguinte informação, consoante os argumentos passados:

- Deve validar os argumentos recebidos e o número de argumentos, podendo em caso de erro dar os erros **error 2** ou **error 3** conforme o caso, terminando em caso de erro.
- Se receber o argumento **listar**, lista o nome de todas as Autoestradas (e.g., "A1", "A23", etc.) que tenham Lanços registados na plataforma (no ficheiro **portagens.txt**), sem repetições (usando o script **success 6**);
- Se receber os argumentos **registos <nr_registos>**, valida se **<nr_registos>** existe (se não existir, dá **error 2**), e se é um número positivo maior que zero (se não for, dá **error 3**). Em caso de erro, termina. Caso contrário, lista o nome de todos os **Lanços** que tenham um número de utilizações registadas no ficheiro **relatorio_utilizacao.txt** maior ou igual a **<nr_registos>** (usando o script **success 6**);

Se receber o argumento **condutores**, lista o nome de todos os condutores com veículos registados na plataforma que tenham passado portagens (ou seja, que tenham registos no ficheiro **relatorio_utilizacao.txt**), usando o script **success 6**.

5) menu.sh

Este script agrega os scripts restantes, mostrando (aqui pode usar **echo**) um menu com as opções:

```
1. Listar condutores
2. Altera taxa de portagem
3. Stats
4. Faturação
0. Sair

Opção: █
```

De seguida, aceita como input do utilizador um número, seguido de <ENTER>. Caso o valor lido seja inválido, deve dar o **error 3**, não invoca nenhum script, e volta a imprimir o menu, retornando ao início. Cada uma das opções anteriores irá invocar o respetivo script descrito nas alíneas anteriores. No caso das opções **2** e **3**, este script deverá pedir interactivamente ao utilizador as informações necessárias para execução do script correspondente, injetando as mesmas como argumentos desse script.

Assim sendo, no caso da opção **2**, o menu deverá pedir ao utilizador sucessivamente os dados a inserir:

```
1. Listar condutores
2. Altera taxa de portagem
3. Stats
4. Faturação
0. Sair

Opção: 2

Altera taxa de portagem...

Lanço           : Lisboa-Porto
Auto-estrada    : A1
Novo valor taxa : 12█
```

Neste caso, o script deverá receber os inputs de Lanço (e.g., “Lisboa-Porto”), Autoestrada (e.g., “A1”) e Novo valor da taxa (e.g., 12), e com estes inputs, invocar o script correspondente, sem ser necessário efetuar qualquer validação dos dados inseridos.

Após a execução de cada subscript, o menu deverá voltar a ser apresentado, e nova opção pedida. Apenas a opção **0** (zero) permite sair deste script “menu”. Até escolher esta opção, o menu deverá ficar em ciclo, permitindo realizar múltiplas operações **iterativamente** (e não recursivamente!).

No caso particular da opção **3**, o script deverá apresentar um novo sub-menu:

```
1. Listar condutores
2. Altera taxa de portagem
3. Stats
4. Faturação
0. Sair

Opção: 3

Stats

1. Nome de todas as Autoestradas
2. Registos de utilização
3. Listagem condutores
0. Voltar

Opção: █
```

Após apresentar o novo sub-menu, o utilizador poderá usar a opção **0** para voltar para o menu anterior (na realidade, significa “não faz nada e volta ao início do script, mostrando novamente o menu inicial”), ou pode introduzir novamente um número seguido de <ENTER> para escolher uma das novas subopções, que irão resultar na invocação do script **stats.sh** com o argumento associado à subopção assinalada pelo utilizador. É claro que quando a subopção **2** for selecionada, o script tem de pedir ainda mais uma informação necessária para invocar:

```
Opção: 2
Mínimo de registos : 14█
```

Anexo

Scripts fornecidos, com Mensagens de **sucesso**, **erro**, **debug** e **validação de Scripts**:

Mensagens de output com Erro (com exemplos): script error <argumentos>

- O ficheiro "ficheiro.txt" não existe:
 - `$./error 1 "ficheiro.txt"`
- Não foram passados argumentos suficientes:
 - `$./error 2`
- O formato do argumento "Taxa" não é o esperado:
 - `$./error 3 "Taxa"`

Mensagens de output com Sucesso (com exemplos): script success <argumentos>

- Mensagem genérica de sucesso por terminar o script sem erros:
 - `$./success 1`
- Mostra a lista de condutores <ficheiro>: Mostra a lista de condutores registados que está no ficheiro (sem fazer qualquer ordenação ou manipulação do ficheiro):
 - `$./success 2 "condutores.txt"` ou então
 - `$ cat "condutores.txt" | ./success 2`
- Mostra mensagem de sucesso na atualização de taxa de utilização de um Lanço:
 - `$./success 3 "Lisboa-Sintra"`
- Mostra a Lista de Portagens <ficheiro>: Mostra a lista de portagens que está no ficheiro (sem fazer qualquer ordenação ou manipulação do ficheiro):
 - `$./success 4 "portagens-ordenadas.txt"` ou então
 - `$ cat "portagens-ordenadas.txt" | ./success 4`
- Mostra a lista de Faturas <ficheiro>: Mostra a lista de faturas que está no ficheiro (sem fazer qualquer ordenação ou manipulação do ficheiro):
 - `$./success 5 "faturas.txt"` ou então
 - `$ cat "faturas.txt" | ./success 5`
- Mostra a lista de Nomes (de autoestradas, ou de lanços, ou de condutores) <ficheiro>: Mostra a lista de nomes que está no ficheiro (sem fazer qualquer ordenação ou manipulação do ficheiro):
 - `$./success 6 "nomes.txt"` ou então
 - `$ cat "nomes.txt" | ./success 6`

Mensagens de Debug: Apesar de não ser necessário, disponibilizou-se também um Shell script para as mensagens de debug dos scripts, dado que será muito útil aos alunos: **script debug <argumentos>**

A utilização nos scripts é, e.g., **var1=3 && ./debug var1=\$var1 # mostra "DEBUG: var1=3"**

Tem a vantagem de que mostra sempre as mensagens de debug (não precisa sequer ser nunca apagado). Quando os alunos quiserem apagar as mensagens de debug, comentam uma linha no Shell script debug, mantendo os vossos scripts intocados.

Script Validador do trabalho:

Como anunciado nas aulas, está disponível para os alunos um script de validação dos trabalhos, para os alunos terem uma noção dos critérios de avaliação utilizados.

Passos para realizar a validação do vosso trabalho:

- Garantam que o vosso trabalho (i.e., os cinco scripts .sh) está localizado numa diretoria local da vossa área. Para os efeitos de exemplo para esta demonstração, assumiremos que essa diretoria terá o nome **trab-so-parte-1** (mas poderá ser outra qualquer);
- Posicionem-se nessa diretoria **trab-so-parte-1** da vossa área:
\$ cd trab-so-parte-1
- Deem o comando **\$ pwd**, e validem que estão mesmo na diretoria correta;
- Deem o comando **\$ ls -l**, e confirmem que todos os ficheiros **.sh** do vosso trabalho estão mesmo nessa diretoria, e que esses ficheiros têm permissão de execução;
- Deem o comando para copiar a diretoria do validador para a vossa diretoria local (.):
\$ cp -r /home/so/trabalho-2021-2022/parte-1/so-2021-trab1-validator/ .
- Deem o comando para que essa diretoria fique com permissões para escrita e execução:
\$ chmod 700 so-2021-trab1-validator/
- Agora, posicionem-se na subdiretoria do validador:
\$ cd so-2021-trab1-validator/
- E, finalmente, executem o script de validação do vosso trabalho, que está na diretoria "pai" (..)
\$./so-2021-trab1-validator.py ..
- Resta agora verificarem quais dos vossos testes "passam" (✓) e quais "chumbam" (✗);
- Façam as alterações para correção dos vossos scripts;
- Sempre que quiserem voltar a fazer nova validação, basta novamente posicionarem-se na subdiretoria **so-2021-trab1-validator** e correrem o script de validação como demonstrado acima.

Boa sorte!!!
Os docentes