

Processamento de Linguagens (3º Ano MIEI)

**Trabalho Prático 1a - GAWK**

Relatório de Desenvolvimento

Fábio Luís Baião da Silva  
(A75662)

João da Cunha Coelho  
(A74859)

Luís Miguel Moreira Fernandes  
(A74748)

13 de Março de 2017

## Resumo

*Regular expressions* can be very useful. This project intends to prove exactly this point, showing how to use them in a subject that makes part of our daily routine.

Via Verde is an electronical toll collection system used in Portugal since 1991. The stock for this project is a monthly extract from one of its users, which will be used to answer some queries using **GAWK**, one of the most prominent text-processing utility on GNU/Linux.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Estrutura do Ficheiro XML</b>	<b>3</b>
<b>3</b>	<b>Apresentação da Solução de Leitura e Parser</b>	<b>4</b>
3.1	Row Separator (RS) . . . . .	4
3.2	Field Separator (FS) . . . . .	4
<b>4</b>	<b>Apresentação das Queries e Resoluções</b>	<b>5</b>
4.1	Queries de Resolução Obrigatória . . . . .	5
4.2	Queries Adicionais . . . . .	6
<b>5</b>	<b>Conclusão</b>	<b>7</b>

# Capítulo 1

## Introdução

O primeiro contacto com as expressões regulares despertou grande curiosidade, dada a capacidade de analisar e interpretar um ficheiro de texto, que pode ser longo, em pouco tempo, recolhendo de uma só vez informação suficiente para responder a várias interrogações. Este trabalho permitirá comprovar essa utilidade e eficiência do **GAWK**, aplicando-o a um exemplo prático do dia-a-dia.

A escolha recaiu no tema da Via Verde devido à clareza do enunciado e à utilidade do tema, uma vez que este tipo de processador de transações pode ser aplicado nas mais diversas áreas.

## Capítulo 2

# Estrutura do Ficheiro XML

De seguida é sucintamente explicada a estrutura do ficheiro XML em análise.

**Cabeçalho com informação sobre o mês a que se refere o extrato e os dados do cliente:**

```
<EXTRACTO id="011114056/08/2015">
<MES_EMISSAO>Ago-2015</MES_EMISSAO>
<CLIENTE id="514714936">
  <NIF>987653210</NIF>
  <NOME>PEDRO MANUEL RANGEL SANTOS HENRIQUES</NOME>
  <MORADA>RUA XXX</MORADA>
  <LOCALIDADE>BRAGA</LOCALIDADE>
  <CODIGO_POSTAL>4715-012 BRAGA</CODIGO_POSTAL>
</CLIENTE>
<IDENTIFICADOR id="28876820811">
<MATRICULA>00-LJ-11</MATRICULA>
<REF_PAGAMENTO>1234567</REF_PAGAMENTO>
```

Todas as transações de serviços oferecidos pela ViaVerde (portagens, parques de estacionamento, etc), sendo que cada transação tem campos como: data e hora de entrada e saída, local de entrada e saída, valor, etc:

```
<TRANSACCAO>
  <DATA_ENTRADA>26-07-2015</DATA_ENTRADA>
  <HORA_ENTRADA>11:33</HORA_ENTRADA>
  <ENTRADA>Povoa N-S</ENTRADA>
  <DATA_SAIDA>26-07-2015</DATA_SAIDA>
  <HORA_SAIDA>11:42</HORA_SAIDA>
  <SAIDA>Angeiras N-S</SAIDA>
  <IMPORTANCIA>2,00</IMPORTANCIA>
  <VALOR_DESCONTO>0,00</VALOR_DESCONTO>
  <TAXA_IVA>23</TAXA_IVA>
  <OPERADOR>I. de Portugal (N1)</OPERADOR>
  <TIPO>Portagens</TIPO>
  <DATA_DEBITO>05-08-2015</DATA_DEBITO>
  <CARTAO>6749036</CARTAO>
</TRANSACCAO>
```

**Rodapé com valores totais:**

```
...
<TOTAL>77,40</TOTAL>
</IDENTIFICADOR>
<TOTAL>77,40</TOTAL>
<TOTAL_IVA>14,49</TOTAL_IVA>
</EXTRACTO>
```

## Capítulo 3

# Apresentação da Solução de Leitura e Parser

```
RS = "<TRANSACCAO>";  
FS = "[<>]";
```

### 3.1 Row Separator (RS)

A opção passou por separar os *records* pela tag `<TRANSACAO>`.

Com o *Record Separator* definido desta forma sabe-se que o primeiro *record* corresponde ao cabeçalho, enquanto que os restantes irão corresponder às transações a analisar.

### 3.2 Field Separator (FS)

Aqui foi tomada a decisão de escolher como *field separator* os caracteres `<e >`. Cada transação é definida por vários segmentos com o seguinte formato:

```
<...>...</...>\n
```

Com o *field separator* que foi anunciado anteriormente, os dados relevantes para análise encontrar-se-ão sempre no segundo (tag) e terceiro (valor) campo de cada linha. Desta forma, para se obter o valor de determinada tag, basta iterar pelos campos de cada linha, obtendo-se a seguinte função:

```
function getValueOf (tag){  
    for (i=2; i < NF; i+=4){  
        if ($i == tag)  
            return $(i+1);  
    }  
    return null;  
}
```

## Capítulo 4

# Apresentação das Queries e Resoluções

### 4.1 Queries de Resolução Obrigatória

A escolha do **RS** permite-nos encontrar, para cada transação, o valor do campo que pretendemos, invocando somente a função *getValueOf*, sendo o argumento esse mesmo campo.

- a) Calcular o número de 'entradas' em cada dia do mês

Para além do número de entradas, extendemos o âmbito da *query* de modo a distinguir as entradas nos diferentes serviços - portagens ou parques de estacionamento.

```
tipo = getValueOf("TIPO");
data = getValueOf("DATA_ENTRADA");
if (tipo != null && data != null && data != "null"){
    invData = inverter(data);
    nEntradas[tipo][invData]++;
}
```

Para a impressão dos resultados foi criada uma tabela *html*, porém aqui mostrar-se-á o processo de impressão no terminal para que seja mais fácil explicar o processo. Começou-se por separar os resultados por tipo. Em seguida, como se pretendia a impressão ordenada das datas, utilizou-se a função *asorti* para criar um array com as datas ordenadas de forma crescente, iterando-o depois no momento de imprimir o valor do número de entradas.

```
for (i in nEntradas){
    print i;
    n = asorti(nEntradas[i], ordenado);
    for (j=1; j <= n; j++){
        data = ordenado[j];
        print inverter(data), nEntradas[i][data];
    }
}
```

- b) Escrever a lista de locais de saída

Como *query* adicional, foi acrescentado a cada local o número de vezes que este foi utilizado como local de saída, bem como o tipo de serviço: **saídas[tipo][saída]** representa o número de vezes que a saída *saída* foi usada para o serviço *tipo* (Portagem ou Parque de Estacionamento).

```

saida = getValueOf("SAIDA");
if (tipo != null && saida != null){
    saidas[tipo][saida]++;
}

for (i in saidas){
    n = asort(saidas[i], ordenado);
    for (j=n; j > 0; j--){
        numeroSaidas = ordenado[j];
        for (k in saidas[i]){
            if (saidas[i][k] == numeroSaidas){
                print k, numeroSaidas;
                delete saidas[i][k];
            }
        }
    }
}
}

```

c) Calcular o total gasto no mês

```

imp = getValueOf("IMPORTANCIA");
desc = getValueOf("VALOR_DESCONTO");
if (imp != null && desc != null){
    gsub(",", ".", imp);
    total += imp - desc;
}

```

```

print values, "TOTAL", total;

```

d) Calcular o total gasto no mês apenas em parques

```

if (tipo != null && imp != null) {
    tipos[tipo] += imp;
}

```

```

n = asort(tipos, ordenado);
for (i=n; i > 0; i--){
    valor = ordenado[i];
    for (j in tipos){
        if (tipos[j] == valor){
            print j, valor;
            delete tipos[j];
        }
    }
}
}

```

## 4.2 Queries Adicionais

a) Cenas fixas

Ccoisas lindas de cenas fixas

b) Mais cenas fixas

Cenas fixas dentro de mais cenas fixas



## Capítulo 5

# Conclusão

Concluído o projeto, destaca-se a facilidade que o filtro de texto permite na recolha de informações do ficheiro. Após a escolha dos **RS** e **FS** apropriados, tornou-se simples responder às *queries* que nos iam surgindo ao longo do desenvolvimento do projeto.

Por outro lado, sublinha-se a criação de uma página *HTML* para a apresentação dos dados, pois tratou-se da nossa primeira experiência com *HTML* e com resultados, a nosso ver, bastante apelativos.

Quanto a trabalho futuro, a interpretação de um ficheiro com extratos de vários meses seria o passo seguinte lógico deste tipo de projeto.