



UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO DE ENGENHARIA INFORMÁTICA

PROCESSAMENTO DE LINGUAGENS

---

## Trabalho Prático 1: AWK

Adriana Guedes

A74545

Bruno Ferreira

A74155

Marco Barbosa

A75278

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Estrutura do Relatório</b>	<b>3</b>
<b>3</b>	<b>Processador de transações da Via Verde</b>	<b>4</b>
3.1	Análise e Especificação . . . . .	4
3.1.1	Descrição informal do problema . . . . .	4
3.1.2	Análise do ficheiro <i>viaverde.xml</i> . . . . .	4
3.1.3	Especificação do Requisitos . . . . .	5
3.2	Conceção/Desenho da Resolução . . . . .	6
3.2.1	Calcular o número de 'entradas' em cada dia do mês. . . . .	6
3.2.2	Escrever a lista de locais de 'saída'. . . . .	6
3.2.3	Calcular o total gasto no mês. . . . .	7
3.2.4	Calcular o total gasto no mês apenas em 'parques'. . . . .	8
3.3	Exemplificação do Programa Gerado . . . . .	10
<b>4</b>	<b>Autores Musicais</b>	<b>12</b>
4.1	Análise e Especificação . . . . .	12
4.1.1	Descrição informal do problema . . . . .	12
4.1.2	Análise de um exemplo de ficheiro .lyr . . . . .	13
4.1.3	Especificação de Requisitos . . . . .	13
4.2	Conceção/Desenho da Resolução . . . . .	14
4.2.1	Calcular o total de cantores e a lista com seus nomes. . . . .	14
4.2.2	Calcular o total de canções do mesmo autor (mesmo que em alguns casos sejam várias pessoas considere como único) . . . . .	15
4.2.3	Escrever o nome de cada autor seguido do título das suas canções; se mais do que uma, separadas por uma vírgula. . . . .	16
4.3	Exemplificação do Programa Gerado . . . . .	17
<b>5</b>	<b>Conclusão</b>	<b>20</b>

# 1 | Introdução

Este primeiro trabalho prático tem como principal objetivo o desenvolvimento, a partir de expressões regulares, de Processadores de Linguagens Regulares, que filtrem ou transformem textos, utilizando o sistema de produção para filtragem de texto GWAK. Gawk é a versão GNU do programa awk UNIX comumente disponível. A função básica do gawk é procurar arquivos por linhas ou outras unidades de texto contendo um ou mais padrões.

Foram disponibilizados pelos docentes da disciplina quatro enunciados, dando-nos a possibilidade de escolher qual deles queríamos desenvolver. Optamos inicialmente pela realização no enunciado **2.1 Processador de transações da Via Verde**, mas após o término deste resolvemos também o enunciado **2.3 Autores Musicais**,

## 2 | Estrutura do Relatório

Este relatório estará dividido em seis partes:

- A primeira parte corresponde à introdução, que irá retratar qual o objetivo do trabalho bem como a escolha dos enunciados a desenvolver.
- A segunda parte corresponde ao presente tópico onde será explicado como está organizado este documento, referindo os capítulos existentes e explicando o conteúdo de cada um.
- A terceira parte corresponde à análise e especificação e, é neste capítulo que é feita uma análise detalhada do problema proposto de modo a se poder especificar as entradas, resultados e formas de transformação.
- Na quarta parte é explicado como foram desenvolvidos os enunciados, bem como a explicação dos algoritmos utilizados.
- Na quinta parte serão apresentados exemplos do programa através de *printscreens*.
- Na ultima parte, sexto capítulo, irá constar a conclusão que irá conter uma síntese do que foi retratado ao longo do documento, as conclusões e o trabalho futuro a ser realizado.

## 3 | Processador de transações da Via Verde

### 3.1 Análise e Especificação

#### 3.1.1 Descrição informal do problema

Neste enunciado foi-nos disponibilizado um ficheiro *.xml* que tivemos que analisar para posteriormente podermos desenvolver um programa apto a lidar com ele. Em tal ficheiro consta um extrato mensal de transações de um identificador *Via Verde* (que diz respeito a um dado cliente), a partir do qual será necessário extrair certas informações. Como referido, tal extrato encontra-se formatado em *XML*, daí que, para podermos obter conteúdo realmente útil e de interesse ao problema, será necessário que o programa desenvolvido seja também capaz de saber lidar e filtrar este tipo de linguagem.

#### 3.1.2 Análise do ficheiro *viaverde.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<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>
    <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>
  </IDENTIFICADOR>
</EXTRACTO>
```

Diagrama de anotações no código XML:

- Uma seta horizontal aponta de `<MES_EMISSAO>Ago-2015</MES_EMISSAO>` para a etiqueta **Mês de faturação**.
- Um retângulo engloba o bloco `<CLIENTE id="514714936">` e a etiqueta **(I)** está à sua direita.
- Outro retângulo engloba o bloco `<TRANSACCAO>` e a etiqueta **(II)** está à sua direita.

Figura 3.1: Excerto do ficheiro utilizado

Como podemos verificar no excerto de ficheiro acima apresentado, existem alguns padrões presentes no documento aos quais nos serão úteis para chegar à solução pretendida.

Assim, temos em (I) as informações relativas ao cliente, no início do ficheiro. Podemos encontrar o NIF, nome, morada, localidade e código postal. Em (II) é a informação relativa às transações, tais como, data de entrada, hora de entrada, local de saída, importância, etc... Este ficheiro apresenta várias transações, todas no mesmo formato.

### 3.1.3 Especificação do Requisitos

Aquando do desenvolvimento do programa capaz de trabalhar com o ficheiro *XML* proposto, qualquer esforço terá que ser principalmente direcionado para a conclusão dos objetivos enunciados, que são encontrar solução para os seguintes quatro pontos:

- calcular o número de 'entradas' em cada dia do mês.
- escrever a lista de locais de 'saída'.
- calcular o total gasto no mês.
- calcular o total gasto no mês apenas em 'parques'.

À medida que fomos trabalhando, decidimos melhorar um pouco este problema e, por isso, tomámos a liberdade de alterar um pouco a apresentação das respostas a cada um dos pontos referidos. Por outras palavras, ao invés das respostas serem tipicamente apresentadas num terminal (como é típico do *AWK*), elas passarão a ser apresentadas numa página *html*, o que não só embeleza o resultado final mas também nos levou a aplicar conhecimentos falados nas últimas aulas práticas, nomeadamente a parte do *html*.

## 3.2 Conceção/Desenho da Resolução

### 3.2.1 Calcular o número de 'entradas' em cada dia do mês.

Para responder a esta pergunta, será necessário encontrar no ficheiro cada campo `<DATA_ENTRADA>#uma data</DATA_ENTRADA>` e, contabilizá-lo como uma entrada. Porém, após uma análise cuidada do ficheiro, notámos que alguns campos apresentam-se na forma `<DATA_ENTRADA>null</DATA_ENTRADA>` mas decidimos contá-los também como uma entrada válida uma vez que o valor *null* possivelmente será devido a algum erro aquando a leitura do identificador na entrada do pórtico. Consideremos o seguinte código desenvolvido:

```
BEGIN{
    PROCINFO["sorted_in"] = "@ind_str_asc"
    fmt = "<li><b>%s:</b> %s</li>\n"
    q1 = "Testes/q1.html"
    print "<p><i><a href='indice.html'>Voltar</a></i></p>" > q1
}

match($0, /<DATA_ENTRADA>(.*?)</DATA_ENTRADA>/, entradas){
    dias[entradas[1]]++
}

END {
    print (" Data      Nr Entradas<br />") > q1
    for (dia in dias)
        printf ("%s:      %s<br />", dia, dias[dia]) > q1
}
```

Como podemos ver, o seguinte código contém a típica estrutura de um programa *AWK*:

- inicialmente existe um bloco *BEGIN* com instruções que são executadas antes de serem lidas quaisquer linhas do ficheiro *viaverde.xml*. Neste bloco, basicamente, existem instruções necessárias e referentes à parte do *html*, e não à pergunta em causa.
- Depois do bloco *BEGIN* segue-se o corpo do programa *AWK*, parte essa que trata especificamente de gerar a resposta à pergunta em causa. Nessa primeira linha deste bloco procuramos em todas as linhas do ficheiro *xml* a expressão regular que contém informação acerca das entradas efetuadas (`/<DATA_ENTRADA>(.*?)<DATA_ENTRADA>/`) e guardámo-las, na terceira linha, num array *dias*. Note-se que esse array vai conter todas as datas de entrada.
- Por fim, o programa termina num *END block* que é executado após todas as linhas do ficheiro *viaverde.xml* terem sido lidas e, por sua vez, o array *dias* contiver todas as datas de entrada existentes no extrato. A primeira linha deste bloco diz respeito ao *html* e é meramente por questões de estética, enquanto que na segunda e terceira linha, procuramos percorrer todas as datas do array e, para cada uma delas, apresentamo-la e apresentamos também o número de entradas que houve.

### 3.2.2 Escrever a lista de locais de 'saída'.

Para responder a esta pergunta, será necessário encontrar no ficheiro cada campo `/<SAIDA>#um local de saída<SAIDA>/` e guardá-la. Para além disso, contabilizámos também, para cada local de saída, o número de vezes ele consta no extrato. Consideremos o seguinte código desenvolvido:

```
BEGIN{
```

```

PROCINFO["sorted_in"] = "@ind_str_asc"
fmt = "<li><b>%s:</b> %s</li>\n"
q2= "Testes/q2.html"
print "<p><i><a href='indice.html'>Voltar</a></i></p>" > q2
}

match ($0 , /<SAIDA>(.*?)</SAIDA>/, locais) {
    local[locais[1]]++
}

END{
    for (l in local)
        printf ("%s : %s<br />", l , local[l]) > q2
}

```

À semelhança da implementação para a primeira pergunta, este programa contém também 3 blocos:

- No bloco *BEGIN* encontram-se, tal como no código anterior, instruções necessárias e referentes à parte do *html*, e não à pergunta em causa.
- No corpo do programa *AWK*, encontra-se a parte essa que trata especificamente de gerar a resposta à pergunta em causa. Nessa primeira linha deste bloco procuramos em todas as linhas do ficheiro *xml* a expressão regular que contém informação acerca dos locais de saída (*/<SAIDA>(.\*?)<SAIDA>/*) e guardamo-los, na terceira linha, num array *local*.
- Por fim, o programa termina num *END block* que percorre todos os locais de saída contidos no array *local* e, para cada um deles, apresentamo-lo e apresentamos também o número de saídas que houve nesse local.

### 3.2.3 Calcular o total gasto no mês.

Para responder a esta pergunta, será necessário encontrar no ficheiro cada campo *<IMPORTANCIA>#um valor</IMPORTANCIA>* e guardá-lo. Após todos os valores lidos, procuramos soma-los todos, obtendo assim o valor correspondente a todas as transações efetuadas com o serviço *Via Verde*.

Consideremos o seguinte código desenvolvido:

```

BEGIN {
    PROCINFO["sorted_in"] = "@ind_str_asc"
    fmt = "<li><b>%s:</b> %s</li>\n"
    q3 = "Testes/q3.html"
    print "<p><i><a href='indice.html'>Voltar</a></i></p>" > q3
}

match ($0 , /<IMPORTANCIA>(.*?)</IMPORTANCIA>/, aux) {
    sub(/,/,".", aux[1])
    total += aux[1]
}

END {
    printf ("O total gasto foi %s<br />", total) > q3
}

```

À semelhança da implementação para a primeira pergunta, este programa contém também 3 blocos:



- No bloco *BEGIN* encontram-se, tal como no código anterior, instruções necessárias e referentes à parte do *html*, e não à pergunta em causa.
- No corpo do programa *AWK*, encontra-se a parte essa que trata especificamente de gerar a resposta à pergunta em causa. Nessa primeira linha deste bloco procuramos em todas as linhas do ficheiro *xml* a expressão regular que contém informação acerca da importância a pagar em cada passagem (`/<IMPORTANCIA>(.*?)<IMPORTANCIA>/`), guardando o valor num array temporário *aux*, substituímos a vírgula de cada um desses valor por um ponto para que seja possível efetuar operações com eles e, na terceira linha deste bloco, atribuímos à variável *total* a soma de todos os valores que se encontram no *aux*.
- Por fim, o programa termina num *END block* que simplesmente trata de imprimir o valor de *total*.

### 3.2.4 Calcular o total gasto no mês apenas em 'parques'.

Para responder a esta pergunta, será necessário encontrar no ficheiro cada campo `<IMPORTANCIA>#um valor</IMPORTANCIA>` especificado com `<TIPO>Parques de estacionamento</TIPO>` e guardá-lo. Após todos os valores lidos, procuramos soma-los todos, obtendo assim o valor correspondente a todas as transações efetuadas com o serviço *Via Verde*.

Consideremos o seguinte código desenvolvido:

```
BEGIN {
    PROCINFO["sorted_in"] = "@ind_str_asc"
    fmt = "<li><b>%s:</b> %s</li>\n"
    q4 = "Testes/q4.html"
    print "<p><i><a href='indice.html'>Voltar</a></i></p>" > q4
}

match ($0 , /<TIPO>(.*?)</TIPO>/, auxtipo) {
    tipo[i] = auxtipo[1]
    i++
}

match ($0 , /<IMPORTANCIA>(.*?)</IMPORTANCIA>/, auxcusto) {
    sub(/,/,".", auxcusto[1])
    custo[i] = auxcusto[1]
}

END{
    for (j = 0 ; j <= i ; j ++ ){
        if (tipo[j] == "Parques de estacionamento")
            totalp += custo[j]
    }
    printf ("O total gasto em parques foi %s<br />", totalp) > q4
}
```

À semelhança da implementação para a primeira pergunta, este programa contém também 3 blocos:

- No bloco *BEGIN* encontram-se, tal como no código anterior, instruções necessárias e referentes à parte do *html*, e não à pergunta em causa.
- No corpo do programa *AWK*, encontra-se a parte essa que trata especificamente de gerar a resposta à pergunta em causa. Na primeira função *match* do *AWK* procuramos em todas as linhas do ficheiro *xml* a expressão regular que contém informação acerca do tipo de serviço que recorreu à *Via Verde* (`/<TIPO>(.*?)<TIPO>/`), guardando num array *tipo* cada um dos tipos de transação existente no extrato. Por outro lado utilizamos outra função *match*

do *AWK* que procura em todas as linhas do ficheiro *xml* a expressão regular que contém informação acerca da importância a pagar em cada tipo de serviço *Via Verde* (`/<IMPORTANCIA>(.*<IMPORTANCIA>/`), guardando o valor num array temporário *auxcusto*, substituímos a vírgula de cada um desses valor por um ponto para que seja possível efetuar operações com eles e depois guardamos esses valores num array *custo*, que usa o mesmo índice '*i*' que o array referente aos tipos de transação.

- Por fim, o programa termina num *END block* que faz uso de um ciclo para percorrer os dois arrays criados anteriormente e, depois de verificar que o tipo corresponde a *Parques de estacionamento*, vai somando o valor corresponde que está no array *custo* a uma variável *totalp* (iniciada a 0), imprimindo-a no fim.

### 3.3 Exemplificação do Programa Gerado

Extracto Mensal ViaVerde	
Ago-2015	
Referente à matrícula 00-LJ-11.	
Referencia pagamento: 1234567	
Nome: PEDRO MANUEL RANGEL SANTOS HENRIQUES	
NIF: 987653210	
Morada: RUA XXX, BRAGA, 4715-012 BRAGA	
Consulte:	
<ul style="list-style-type: none"><li>• <a href="#">Número de entradas em cada dia do mês</a></li><li>• <a href="#">Lista de locais de saída</a></li><li>• <a href="#">Total gasto no mês</a></li><li>• <a href="#">Total gasto no mês apenas em parques de estacionamento</a></li></ul>	

#### Numero de entradas referentes a cada dia do mês

Data	Numero de Entradas
06-08-2015	4
10-08-2015	7
11-08-2015	2
13-08-2015	5
17-08-2015	4
18-08-2015	2
21-08-2015	4
26-07-2015	3
29-07-2015	2
30-07-2015	3
31-07-2015	1

#### Locais de saída

Local	Numero de Saídas
Aeroporto	1
Angeiras N-S	5
Braga Sul	2
Custoias	1
EN 205 PV	6
EN107	1
Ermesinde PV	1
Ferreiros	6
Freixiello	2
Lipor	1
Maia II	1
Maia PV	1
Neiva N-S	1

[Voltar](#)

**Total gasto referente ao mês Ago-2015**

O total gasto no mês apresentado foi de 77.4€

[Voltar](#)

**Total gasto em parques referente ao mês Ago-2015**

O total gasto no mês apresentado em parques foi de 6.35€

## 4 | Autores Musicais

### 4.1 Análise e Especificação

#### 4.1.1 Descrição informal do problema

Depois de termos concluído o primeiro enunciado, decidimos investir um pouco mais neste trabalho e tentamos desenvolver o terceiro enunciado também, como complemento extra.

Neste enunciado em particular é disponibilizada uma diretoria denominada de *musica* (anexada em formato ZIP), que contém uma panóplia de ficheiros com extensão *.lyr*. Cada um desses ficheiros contem informações referentes a uma música específica e, no final, pretende-se a criação de um programa capaz de lidar, filtrar e apresentar estes ficheiros com uma dada disposição.

### 4.1.2 Análise de um exemplo de ficheiro .lyr

```
title: Pronúncia do Norte
lyrics: Rui Reininho
music: Tóli César Machado
singer: GNR; Isabel Silvestre
in: "rock in rio Douro", 1992
from: A. Guimarães

Há um prenúncio de morte
Lá do fundo de onde eu venho
Os antigos chamam-lhe renho
Novos ricos são má sorte

É a pronúncia do Norte
Os tontos chamam-lhe torpe

Hemisfério fraco outro forte
Meio-dia não sejas triste
A bússula não sei se existe
E o plano talvez aborte

Nem guerra, bairro ou corte
É a pronúncia do Norte

Não tenho barqueiro nem hei-de remar
Procuro caminhos novos para andar
Tolheste os ramos onde pousavam
Da Geada as pérolas as fontes secaram

Corre um rio para o mar
E há um prenúncio de morte

E as teias que vidram nas janelas
esperam um barco parecido com elas
Não tenho barqueiro nem hei-de remar
Procuro caminhos novos para andar

E É a pronúncia do Norte
Corre um rio para o mar
```

Figura 4.1: Excerto do ficheiro utilizado

Neste excerto podemos ver que na secção (I) do ficheiro encontram-se informações relativas à canção (como o título, informações do cantor, álbum, etc) enquanto que na secção (II) está contida a própria letra da música. Note-se porém que, a letra da música pode conter alguns caracteres que não fazem parte dela e que, portanto, terão que ser removidos.

### 4.1.3 Especificação de Requisitos

O desenvolvimento do programa *AWK* para este enunciado deverá ser capaz de apresentar resposta às seguintes questões:

- calcular o total de cantores e a lista com seus nomes.
- calcular o total de canções do mesmo autor (mesmo que em alguns casos sejam várias pessoas considere como único).
- escrever o nome de cada autor seguido do título das suas canções; se mais do que uma, separadas por uma vírgula.

## 4.2 Conceção/Desenho da Resolução

Neste capítulo, procurámos explicar todos os passos tomados na resolução do enunciado **2.3 Autores Musicais**, sendo que para isso apresentaremos/explicaremos toda a codificação que permite a resposta a cada uma das perguntas deste problema.

Assim, passaremos a enunciar cada pergunta do problema, e a codificação correspondente:

### 4.2.1 Calcular o total de cantores e a lista com seus nomes.

Para responder a esta pergunta, será necessário encontrar em todos os ficheiros *.lyr* um campo *singer*, caso exista, e analisar o seu conteúdo, removendo quaisquer formatações não necessárias. Consideremos o seguinte código desenvolvido:

```
BEGIN {
PROCINFO["sorted_in"] = "@ind_str_asc";
FS = ":"
}

function norm(str) {
    return tolower(str);
}

{
    if (norm($1) == "singer") {
        if ($2 != " ?"){
            gsub(",",";")
            x = split($2, a, ";")
            for (j in a) {
                gsub("&","e",a[j])
                gsub(/ .$/,"",a[j])
                gsub(/ ^ /,"",a[j])
                gsub(/ $/,"",a[j])
                cant[a[j]]}
        }
    }
}

END {
    for (c in cant)
        print "Cantor -> " c

    y = length(cant)

    printf ("No total são %s cantores.\n" , y )
}
```

Este programa desenvolvido contém as seguintes partes:

- No bloco *BEGIN*, essencialmente, definimos o *Field Separator* do *AWK* como sendo ":".

- Depois existe uma função *norm* que trata apenas de passar para caracteres minúsculos todos os caracteres de uma dada string
- No corpo do programa *AWK*, encontra-se a parte essa que trata especificamente de gerar a resposta à pergunta em causa. A primeira função é executada em todas as linhas de todos os ficheiros, e para aquelas onde o primeiro campo (\$1) é igual a *singer*, ao segundo campo é aplicada a função *split* que irá guardar no array *a* os respetivos campos, que estão separados por ";". De seguida é aplicada a função *a* a todos os elementos do array *a*. Esta será aplicada várias vezes a fim de fazer um "tratamento" ao texto de forma a que possamos depois juntar os dados e obtermos os resultados certos. Após isso, cada elemento do array *a* será guardado no array *cant*, em que as chaves deste serão os próprios elementos.
- Por fim, o programa termina num *END block* que trata de percorrer um array *cant* que contém todos os cantores e, para cada um, imprime-os. Depois é calculado também o tamanho do array referido para que depois possa ser imprimido numa mensagem.

#### 4.2.2 Calcular o total de canções do mesmo autor (mesmo que em alguns casos sejam várias pessoas considere como único)

Para responder a esta pergunta, será necessário encontrar em todos os ficheiros.lyr um campo *author*, caso exista, e analisar o seu conteúdo, removendo quaisquer formatações não necessárias. Consideremos o seguinte código desenvolvido:

```
BEGIN {
PROCINFO["sorted_in"] = "@ind_str_asc";
FS = ":"
}

function norm(str) {
    return tolower(str);
}

{
if (norm($1) == "author"){
gsub(/^ /,"",$2)
gsub(/ *$/,"",$2)
gsub("; ", ";",$2)
gsub(" ;", ";",$2)
autores[$2]++
}
}

END {
for (a in autores) {
printf ("%s -> %s \n ", a , autores[a])
}
}
```

Este programa desenvolvido contém as seguintes partes:

- No bloco *BEGIN*, essencialmente, definimos o *FS* do *AWK* como sendo ":".



- Depois existe uma função *norm* que trata apenas de passar para caracteres minúsculos todos os caracteres de uma dada string.

- No corpo do programa *AWK*, encontra-se a parte que trata especificamente de gerar a resposta à pergunta em causa.

A primeira é aplicada a todas as linhas de todos os ficheiros *.lyr*, e caso o \$1 (primeiro campo) seja igual a "author" será aplicada a função *gsub* ao \$2 (segundo campo, que irá corresponder ao nome de autor). Esta será aplicada várias vezes a fim de fazer um "tratamento" ao texto de forma a que possamos depois juntar os dados e obtermos os resultados certos. Após isso, iremos incrementar 1 unidade aquele autor, que corresponderá a mais uma música sua.

- Por fim, o programa termina num *END* que trata de percorrer um array *autores* que contém todos os autores e, para cada um, imprime-os juntamente com o número de músicas de cada autor.

### 4.2.3 Escrever o nome de cada autor seguido do título das suas canções; se mais do que uma, separadas por uma vírgula.

Para responder a esta pergunta, será necessário analisar todos os ficheiros *.lyr* e, para cada música, associá-la aos seus autores, agrupando-as depois de forma a que o resultado possa ser imprimido corretamente. Consideremos o seguinte código desenvolvido:

```
BEGIN{
    FS= ":"
    i = 0
    PROCINFO["sorted_in"] = "@ind_str_asc";
}

function norm(str) {
    return tolower(str);
}

{
    if (norm($1) == "title") {
        titles[i] = $2
        gsub(" ", "", titles[i])
        gsub(/\^ /, "", titles[i])
        gsub(/ $/, "", titles[i])
    }
}

{
    if (norm($1) == "author") {
        autores[i] = $2
        gsub(/\^ /, "", autores[i])
        gsub(/ $/, "", autores[i])
        i++
    }
}

END{
```

```

for ( j = 0 ; j < i ; j++)
    cancoes[autores[j]] = titles[j] "," cancoes[autores[j]]

for (c in cancoes) {
    printf ("%s -> %s\n" , c , cancoes[c])}
}

```

Este programa desenvolvido contém as seguintes partes:

- No bloco *BEGIN*, essencialmente, definimos o *Field Separator* do *AWK* como sendo ":".
- Depois existe uma função *norm* que trata apenas de passar para caracteres minúsculos todos os caracteres de uma dada string.
- Caso se trate de um título, ele será guardado num array *titles*, sendo que depois são removidas quaisquer formatações que possam existir através de uma função do *AWK* : *gsub*
- Caso se trate de um autor, ele será guardado num array *autores*, sendo que depois são removidas quaisquer formatações que possam existir através de uma função do *AWK* : *gsub*
- Por fim, o programa termina num *END block* que trata de juntar os elementos de cada um dos arrays num outro array *cancoes*, separados por vírgula, e depois imprime-os.

## 4.3 Exemplificação do Programa Gerado

### Biblioteca de Letras Musicais



A biblioteca aqui presente contém várias letras de canções famosas, maioritariamente de músicas portuguesas. Para que se possa adicionar um ficheiro a esta biblioteca é necessário que o mesmo tenha uma série de características para que se possa fazer a verificação e atualização da biblioteca.

Para tal os ficheiros devem obedecer as seguintes regras:

- A extensão do ficheiro tem de ser '.lyr'.
- A letra da musica é precedida de 2 ou mais campos de meta-informação(1 por linha) com o título da canção, o autor da letra(1 ou mais autores), o cantor, etc.
- A meta-informação é separada da letra por uma linha em branco
- Podemos ter ainda um terceiro bloco, também separado por uma linha em branco, que contém a musica escrita na notação midi

- [Enunciar todos os autores e as suas musicas correspondentes](#)
- [Total de canções de um dado autor](#)
- [Total de cantores e a lista dos seus nomes](#)

[Voltar](#)

## Lista de cantores

No total temos presentes 128 cantores nesta biblioteca.

- Samuel
- A. P. Braga
- Adriana Calcanhoto
- Adriano Correia de Oliveira
- Adriano Correia de Oliveira (?)
- Ala dos Namorados
- Alberto Ribeiro
- Alcoolémia
- Alma Lusa
- Amália Rodrigues
- António Calvário
- António Menano
- António Variações
- Banda do Casaco
- Bandemónio
- Bando dos Gambozinos

[Voltar](#)

## Lista de canções de um dado autor

Se um dado autor contiver mais do que uma musica, estas serão apresentadas seperadas por vírgulas.

?

Um caso mais, O anzol, Mestre de culinária, A lua nasceu, Não voltarei a ser fiel, Olha o rio que vai correndo, Da cor do pecado, Neve, Júlia Florista, O porquinho, O indiozinho, = Era uma vez um cuco que não gostava de couves, O cuco, Quase tudo, Vídeo Maria, Cais, Me Faz Bem, Lua-de-Mel (Como o Diabo Gosta), Chuva de Prata, Protesto, Vira de Coimbra, Torre de Santa Cruz, Os teus olhos são tão verdes, O sol anda lá no céu, O meu menino, O meu desejo, Feiticeira, Esmeralda verde, Coimbra menina e moça, Coimbra, Balada do Mondego, Balada da despedida, Balada da despedida (2), Ao morrer os olhos dizem, Ao cair da tarde, A minha capa rasgada, À meia noite ao luar, Adeus Sé velha, Modinha, Filhos da Nação, De 2.ª a 6.ª Feira, Não sei se mereço,

**A. Amargo;A. Duarte**

Há festa na Mouraria,

**A. P. Braga**

Canção para desfazer equívocos,

## Lista de autores

**E o total de canções presentes do dado autor.**

? tem presentes 42 musicas

**A. Amargo;A. Duarte** tem presentes 1 musicas

**A. P. Braga** tem presentes 1 musicas

**Alberto Janes** tem presentes 1 musicas

**Alexandre** tem presentes 1 musicas

**Alfredo Duarte;Amália Rodrigues** tem presentes 1 musicas

**Alves Coelho** tem presentes 1 musicas

**Amadeu do Vale;Frederico Valério** tem presentes 1 musicas

**António Luís Melo;João Bastos** tem presentes 1 musicas

## 5 | Conclusão

A realização deste trabalho prático, o primeiro desta unidade curricular, permitiu melhorar a capacidade de escrita de uma expressão regular (*ER*), e utilizar estas para o desenvolvimento de Processadores de Linguagens Regulares. Para que tal fosse possível, foi necessário aprofundar o nosso conhecimento sobre a linguagem do sistema AWK e assim desenvolver as soluções para os problemas apresentados.

Embora no enunciado ser pedido apenas a elaboração de um enunciado, o nosso grupo desenvolveu dois, e para cada um destes tivemos o cuidado de apresentar os resultados em ficheiros HTML, como forma à sua leitura ser mais agradável e cómoda.

Como conclusão, estamos satisfeitos com o trabalho que desenvolvemos, embora em algumas alíneas não obtivemos os resultados que realmente desejávamos.

Lista-se a seguir o código do programa que foi desenvolvido.