



Processamento de Linguagens

MIEI - 3º ANO - 2º SEMESTRE

UNIVERSIDADE DO MINHO

TRABALHO PRÁTICO Nº2

GAWK



Pedro Freitas
A80975



Francisco Freitas
A81580

28 de Abril de 2019

Conteúdo

1	Contextualização	2
2	Processador de Processos de Emigração	3
2.1	Enunciado	3
2.2	Descrição do problema	3
2.2.1	a)	3
2.2.2	b)	4
2.2.3	c)	4
2.2.4	d)	4
2.3	Decisões e implementação	4
2.3.1	a)	4
2.3.2	b)	5
2.3.3	c)	6
2.3.4	d)	7
2.4	Como obter os resultados	8
2.5	Resultados Obtidos	9
2.5.1	a)	9
2.5.2	b)	10
2.5.3	c)	11
2.5.4	d)	13
3	Apreciação Crítica	15

1. *Contextualização*

Este relatório é o resultado do segundo trabalho prático da Unidade Curricular Processamento de Linguagens. Este trabalho teve por base a utilização do sistema de produção para filtragem de texto GAWK para podermos responder a uma série de alíneas do enunciado.

De todos os enunciados possíveis vamos trabalhar sobre o enunciado **Processador de Processos de Emigração**.

Ao longo do relatório vamos explicar o que nos foi pedido assim como as decisões e abordagens ao enunciado para podermos obter o resultado final pretendido.

2. *Processador de Processos de Emigração*

2.1 Enunciado

Analise com todo o cuidado o ficheiro *natura.di.uminho.pt/jj/pl-19/TP2/emigra.csv* o qual contém informação sobre processos de pedido de passaporte para emigrar, registados no início do sec. XX. Construa então um ou mais programas Awk que processem esse arquivo de modo a:

- a) contar o número de processos registados por Concelho e Freguesia.
- b) calcular a frequência de processos por ano e relacionar com os concelhos.
- c) estudar a ocorrência de nomes próprios (não considere só os requerentes, mas considere também seus parentes).
- d) desenhar um grafo (em DOT) que a partir dos requerentes os relacione com os pais e o cônjuge.

2.2 Descrição do problema

Tal como foi referido anteriormente todo este trabalho baseia-se em utilizar o sistema de filtragem GAWK de forma a conseguirmos obter a informação considerada útil de forma a resolver o problema.

No nosso caso temos de pegar num ficheiro **csv** que contém diversas entradas que contém os valores de dados correspondentes a campos definidos no topo do ficheiro. Neste ficheiro iremos, por exemplo, processar as entradas de forma a encontrar o número de requerentes por freguesia, assim como os nomes de seus parentes e até fazer um grafo em **DOT** que relaciona os familiares. Toda esta informação será organizada e armazenada em ficheiros *HTML* exceto o grafo que além de ser criado o ficheiro *.dot* o resultado também estará num ficheiro **pdf**.

2.2.1 a)

Neste primeiro exercício teríamos de percorrer o ficheiro *csv* e encontrar o **Concelho** e a **Freguesia** de cada entrada e contabiliza-las.

2.2.2 b)

Neste segundo exercício temos de percorrer outra vez o ficheiro *csv* e agora além de procurar o valor de cada entrada relativa ao campo **Concelho** temos de encontrar também o valor do campo **Ano**. Além disto temos de tentar relacioná-los.

2.2.3 c)

Neste exercício voltamos a percorrer o ficheiro *csv* mas neste caso para obtermos as ocorrências de nomes próprios. Estes nomes próprios não são considerados apenas os nomes dos requerentes mas também dos seus familiares (pai, mãe ou cônjuge).

2.2.4 d)

Neste último exercício percorremos mais uma vez o ficheiro *csv*. Agora teremos de associar os requerentes e seus parentes.

2.3 Decisões e implementação

Para resolvermos todos os problemas que nos foram apresentados temos de tomar algumas considerações gerais. Assim, depois de uma análise ao ficheiro *csv* podemos ver que as duas primeiras linhas do ficheiro são compostas pelos os campos do mesmo e os seus separadores. Depois de olhar atentamente podemos concluir que o **FS** é composto pela expressão regular: `;`, permitindo que cada campo possa ser separado por um `;`, isto é que todos diferentes exemplos a seguir podem representar a mesma informação:

CAMPO(\$1);CAMPO(\$2);CAMPO(\$3);CAMPO(\$4);

Devido a características do ficheiro, existem campos que contém valores vazios.

Também decidimos implementar os nossos processadores de forma a escreverem os resultados em ficheiros *html* ou em ficheiro *dot*. Existe por isso um processador em particular (o processador d) que no processo de filtragem escreve diretamente no ficheiro de output. Para facilitar a escrita declaramos uma variável no **BEGIN** que represente esse mesmo ficheiro de output.

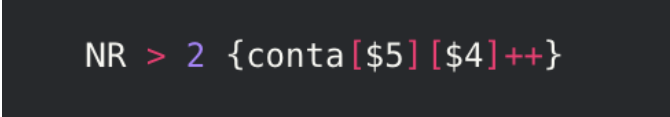
2.3.1 a)

Como foi referido na secção anterior este primeiro exercício baseia-se em analisar o ficheiro de input *csv* de forma a contabilizarmos o número de processos registados em **Concelhos** e **Freguesias**.

Para isso nós entendemos que deveríamos dividir o output em 2 partes. Uma parte continha uma tabela com as diversas freguesias de um dado concelho (cujo seu nome estaria antes da tabela, como título na mesma) e os respetivos números

de entradas. A segunda parte seria a contabilização total por concelho, ou seja, uma nova tabela com todos os concelhos e respetiva contabilização.

Depois de analisar o ficheiro vemos que a **Freguesia** é nos dado no campo número 4 (**\$4**) enquanto que o **Concelho** é nos dado no campo número 5 (**\$5**). Por isso de forma a contabilizar tivemos de optar por uma espécie de matriz que percorria o ficheiro a partir da terceira linha (como já foi referido neste capítulo) e que o primeiro índice era o valor do campo correspondente ao concelho e o segundo índice era o valor do campo correspondente à freguesia. Na seguinte imagem podemos ver iso mesmo:



```
NR > 2 {conta[$5][$4]++}
```

Figura 2.1: Processador-a

Depois da matriz ter sido preenchida e o ficheiro tiver sido todo processado bastou fazer 4 ciclos *for*: dois para a primeira parte do output onde nos interessava escrever no ficheiro de output os dois campos e o respetivo valor; e mais dois para a segunda parte do output onde apenas íamos ao segundo campo da matriz para contabilizar o total de entradas que continham o primeiro índice da matriz.

2.3.2 b)

Para este segundo problema temos de seguir um raciocínio idêntico ao do exercício anterior porém desta vez em vez de analisarmos as freguesias iremos analisar o **Ano** de cada processo e contabilizar de forma análoga ao primeiro exercício, sendo que o **Ano** pode ser encontrado no campo número 6 (**\$6**).

Mais uma vez o ficheiro resultante vai ter duas partes. A primeira parte terá uma tabela por Concelho onde terão cada ano e o número de processos pedidos nesse mesmo ano.

Para isso temos de ver que o campo ano pode ter formas diferentes de se representar e que tem informação não necessária para esta resolução. Depois de analisar o ficheiro de input vemos que a data tem dois formatos diferentes:

ANO-MÊS-DIA

ou:

ANO.MÊS.DIA

Por isso teríamos de arranjar forma de partir esse campo sempre que encontrasse um - ou um . . Assim com a função *split* podemos facilmente obter um array de tamanho 3, onde o primeiro valor do array (de índice 1) é o valor desejado: o ano.

Tendo obtido isso achamos melhor criar 2 arrays para simplificar o processo de escrita no output. Um dos arrays, tal como no exercício anterior, é uma matriz, onde os índices são o concelho e o ano, enquanto que o outro é um array simples onde os índices são os valores dos anos. Sendo esta a decisão o processo de contabilização das entradas é muito trivial:

```
NR > 2 { split($6,a,"[-]|[.]");
        conta[$5][a[1]]++;
        contaA[a[1]]++
}
```

Figura 2.2: Processador b

Mais uma vez depois da matriz ter sido preenchida e o ficheiro tiver sido todo processado bastou fazer 3 ciclos *for*: dois para a primeira parte do output onde nos interessava escrever no ficheiro de output os dois campos e o respetivo valor; e outro para a segunda parte do output.

2.3.3 c)

Neste exercício temos de ver a ocorrência de nomes próprios, não só dos requerentes, mas também dos seus parentes (pai,mãe e cônjuge). Um ponto a ter em consideração é a definição de nomes próprios. Por isso nomes como "Augusto Cruz Gonçalves" têm **três** nomes próprios: "Augusto", "Cruz" e "Gonçalves"; e nomes como "Maria Lourenço do Vale" têm também **três** nomes próprios: "Maria", "Lourenço" e "Vale".

Com isto a nossa abordagem foi verificar os campos dos requerentes, pai e mãe do mesmo e ainda o seu cônjuge, respetivamente nos campos **\$2**, **\$7**, **\$9** e **\$11**. Nesses campos se existir pelo menos uma letra eles separam todo o valor do campo num array, usando a função *split* onde o elemento de separação é o espaço em branco. Depois são percorridas todas as posições do array e caso alguma posição tenha presente um conetor pertencente ao seguinte conjunto: {"de","da","do","das","dos","e"} esse valor é ignorado, caso contrário é incrementado o valor num array que contabiliza todas as entradas desse mesmo nome. Caso em um dos campos não contenha pelo menos uma letra, ou então esteja a expressão "Não tem" o campo é ignorado.

Além de contabilizar a totalidade dos nomes decidimos também contabilizar por campos. Por outras palavras, no output final iremos ver quatro tabelas. Uma com a contagem total, outra com a contagem dos nomes próprios presentes no campo dos requerentes, outra com a contagem dos nomes do pai ou da mãe e ainda uma última tabela com os nomes presentes apenas como cônjuges dos requerentes.

Sabendo isto o resultado codificado será:

```

NR>2 {if ($2 ~ /[A-Z].+ ){
    split($2,a," ");
    for (nome in a)
        if(a[nome]!="de" && a[nome]!="do"&& a[nome]!="dos"&& a[nome]!="da" && a[nome]!="e"&& a[nome]!="")
            conta[a[nome]]++;contaPessoal[a[nome]]++;
    }
    if ($7 != "Não tem" && ($7 ~ /[A-Z].+ )){
        split($7,a," ");
        for (nome in a)
            if(a[nome]!="de" && a[nome]!="do"&& a[nome]!="dos"&& a[nome]!="da" && a[nome]!="e"&& a[nome]!="")
                conta[a[nome]]++;contaPar[a[nome]]++;
    }
    if ($9 ~ /[A-Z].+ ){
        split($9,a," ");
        for (nome in a)
            if(a[nome]!="de" && a[nome]!="do"&& a[nome]!="dos"&& a[nome]!="da" && a[nome]!="e"&& a[nome]!="")
                conta[a[nome]]++;contaPar[a[nome]]++;
    }
    if ($11 ~ /[A-Z].+ ){
        split($11,a," ");
        for (nome in a)
            if(a[nome]!="de" && a[nome]!="do"&& a[nome]!="dos"&& a[nome]!="da" && a[nome]!="e"&& a[nome]!="")
                conta[a[nome]]++;contaConj[a[nome]]++;
    }
}
}

```

Figura 2.3: Processador-c

2.3.4 d)

Neste último exercício além de termos de ssber trabalhar com o sistema de produção de filtragem GAWK, também teríamos de ter uma breve base da ferramenta *DOT* que nos permite criar representação de grafos de uma forma muito simples e imediata.

Nesta alínea era-nos pedido que fosse desenhado um grafo (como dito em cima, em *DOT*) que a partir do requerente o relacionasse com os seus pais e respetivo cônjuge.

Este exercício tal como o anterior trabalhava apenas com os campos **\$2**, **\$7**, **\$9** e **\$11** e também estes não poderiam ser vazios ou iguais a "Não tem" ou seriam ignorados.

Em termos de filtragem apenas era verificado se estes campo continham valores e diferentes da expressão "Não tem" sendo a única novidade perante o exercício anterior a escrita no formato *.dot*. Ainda antes da filtragem dos valores teremos de escrever na primeira linha do ficheiro *.dot* o começo do diagrama e que as setas serão representadas da esquerda para a direita. Estas ações são possíveis com as seguintes expressões, respetivamente:

```

graph LR;

```

Depois disto chegou a hora da filtragem. Neste momento vamos aos campos desejados havendo dois tipos de comportamentos. O primeiro passo é verificar se o campo número 2 contém valores. Se sim vamos verificar se o campo nº7 e nº9 existem e, no caso do campo nº7 se é diferente de "Não tem". Se isto se verificar escrevemos no ficheiro *.dot* o nome do campo respetivo, uma seta e o valor do campo nº2 :

```
$7 -> $2
```


ou:

\$9 -> \$2

, sendo estas setas pintadas de vermelho e contém, respetivamente, a seguinte label: "Pai de"; "Mãe de". Depois da verificação destes dois campos vamos verificar o campo nº11 e se este tem valores. Se sim escrevemos no ficheiro *.dot* a seguinte expressão:

\$2 -> \$11

e agora a cor da seta será azul e a respetiva label: "Cônjuge".

Depois de processar todas as entradas está na hora de fechar terminar a escrita do ficheiro *DOT* fazendo para isso a escrita de : "}".

Além da escrita achamos por bem fazer com o que processador pedisse à máquina para compilar o ficheiro *.dot* convertendo-a em *pdf*.

```
NR >2 {
    if($2 ~ /[A-Za-z]/ ){
        if($7 ~ /[A-Za-z]/ && $7 != "Não tem" )
            print "\x22" $7 "\" -> \"\" $2 "\"[color=red,label=\x22 Pai de \x22];" > local;
        if($9 ~ /[A-Za-z]/ )
            print "\x22" $9 "\" -> \"\" $2 "\"[color=red,label=\x22 Mãe de \x22];" > local;
        if($11 ~ /[A-Za-z]/ )
            print "\x22" $2 "\" -> \"\" $11 "\"[color=blue,label=\x22 Cônjuge\x22];" > local;
    }
}
```

Figura 2.4: Processador-d

2.4 Como obter os resultados

De forma a podermos testar o trabalho final e ver os resultados de toda a implementação realizada teremos de abrir o terminal e ir para a diretoria onde está o projeto: `YOUR_PATH/PL/TP2$` . Dentro dessa diretoria veremos que temos 4 ficheiros de extensão *.awk* e uma Makefile. Para correr também será necessário o ficheiro de extensão *csv* que podemos encontrar em: <http://natura.di.uminho.pt/~jj/pl-19/TP2/emigra.csv>

Testar qualquer um destes exercícios é muito simples. Se quisermos correr todos os processadores basta ter o ficheiro anteriormente referidos na mesma diretoria de todo o projeto e digitar no terminal:

make run

Depois disto verá no terminal que o todos os processadores foram compilados e executados. Além disso terá sido criada uma nova pasta de nome: **output** que contém todos os ficheiros *html* resultantes assim como o ficheiro *dot* e respetivo *pdf*.

Se em vez disso quisermos executar apenas um processador em específico basta fazer:

make processador-X

onde X varia entre **a,b,c,d**. Após isto na pasta **output** irá ser criado ou o *html* correspondente ou o ficheiro *dot* e respetivo *pdf*.

Se quiser apagar os ficheiros resultantes de um exercício basta fazer **make clean-X**, onde mais uma vez X varia entre **a,b,c,d**. Para eliminar todos os executáveis e todos os ficheiros resultantes: **make clean** .

2.5 Resultados Obtidos

Nesta secção poderemos ver os resultados obtidos depois de implementada a nossa solução.

2.5.1 a)

Para o primeiro exercício o output será o ficheiro *ex-a.html* que terá as tais tabelas: uma por concelho e ainda uma com todos os concelhos. Nas duas seguintes imagens poderemos ver, respetivamente, duas tabelas de dois Concelhos e ainda a tabela total:

Ponte da Barca

Freguesia	Número de processos registados
Asias	1

Fare

Freguesia	Número de processos registados
Fareja	1

Figura 2.5: Excerto do ficheiro *ex-a.html* com duas tabelas de Concelhos

Concelhos

Concelho	Número de processos registados
Ponte da Barca	1
Fare	1
Fafe	54
Guimarães	1
Total:	57

Figura 2.6: Excerto do ficheiro *ex-a.html* com a tabela total de Concelhos

2.5.2 b)

Este exercício terá como ficheiro de output o ficheiro *ex-b.html*. Este ficheiro terá uma tabela por Concelho e ainda uma tabela com todo os anos presentes no ficheiro de input. Nas duas seguintes imagens poderemos ver, respetivamente, duas tabelas de dois Concelhos e ainda a tabela total:

Ponte da Barca

Ano	Número de processos registados
1925	1

Fare

Ano	Número de processos registados
1930	1

Figura 2.7: Excerto do ficheiro *ex-b.html* com duas tabelas de Concelhos

Por Ano

Ano	Número de processos registrados
1910	1
1912	1
1917	1
1920	1
1921	1
1922	1
1923	2
1924	3
1925	4

Figura 2.8: Excerto do ficheiro *ex-b.html* com um excerto da tabela total de Anos

2.5.3 c)

Neste exercício temos como resultado o ficheiro *ex-c.html*. Este ficheiro terá 4 tabelas: uma com o total de ocorrências dos nomes, outra com as ocorrências nos requerentes, outro em um dos pais do requerente e ainda outra com o cônjuge do mesmo. Nas quatro seguintes imagens poderemos ver, respetivamente, quatro excertos de cada tabela:

Total

Nome Próprio	Número de ocorrências
Braz	2
Claudino	2
Nogueira	5
Olinda	2
Queirós	1
Albina	2
Soledade	1
Soares	2
Vaz	6

Figura 2.9: Excerto do ficheiro *ex-c.html* com excerto da tabela Total

Requerentes

Nome Próprio	Número de ocorrências
Alves	1
Fernandes	4
Nogueira	1
Pereira	2
Castro	2
Silva	1
Branco	1
Novais	1
Lopes	1

Figura 2.10: Excerto do ficheiro *ex-c.html* com excerto da tabela Requerentes

Parentes de um dos requerentes

Nome Próprio	Número de ocorrências
Braz	1
Nogueira	2
Soares	1
Vaz	1
Delgado	1
Pinto	2
Fernandes	4
Branco	1
Neiva	1

Figura 2.11: Excerto do ficheiro *ex-c.html* com excerto da tabela Parentes

Conjuge de um dos requerentes

Nome Próprio	Número de ocorrências
Fernandes	2
Castro	1
Pereira	1
Nogueira	2
Lourenço	1
Lopes	1
Silva	4
Pinto	1
Mota	1

Figura 2.12: Excerto do ficheiro *ex-c.html* com excerto da tabela Cônjuge

2.5.4 d)

Neste exercício temos como resultado dois ficheiros *arvore.dot* e *arvore.pdf*, que resulta do primeiro ficheiro. Como já foi referido anteriormente, o processador escreve no ficheiro *.dot* e só depois de estar completo é que o processador pede à máquina para o converter em *pdf*. Nas duas imagens seguintes poderemos ver, respetivamente, um excerto do ficheiro *arvore.dot* e o respetivo excerto do ficheiro *arvore.pdf*:

```
"Álvaro Teixeira da Silva e Castro e Júnior" -> "António Alberto Teixeira da Silva e Castro"[color=red,label=" Pai de "];
"Maria Alice Leite Gonçalves" -> "António Alberto Teixeira da Silva e Castro"[color=red,label=" Mãe de "];
"Francisco Martins" -> "Maria de Lourdes Martins"[color=red,label=" Pai de "];
"Rosa Sampaio" -> "Maria de Lourdes Martins"[color=red,label=" Mãe de "];
"Maria de Lourdes Martins" -> "Artur Castro"[color=blue,label=" Cônjuge"];
"David dos Santos Pinheiro" -> "Artur César Pinheiro"[color=red,label=" Pai de "];
"Maria Rosa da Silva" -> "Artur César Pinheiro"[color=red,label=" Mãe de "];
"Artur César Pinheiro" -> "Maria Gonçalves da Silva"[color=blue,label=" Cônjuge"];
"Adriano Gonçalves da Silva" -> "João Gonçalves da Silva"[color=red,label=" Pai de "];
"Rosalina Lopes" -> "João Gonçalves da Silva"[color=red,label=" Mãe de "];
"João Gonçalves da Silva" -> "Deolinda Gonçalves da Silva"[color=blue,label=" Cônjuge"];
"Angelino de Freitas" -> "Amélia de Freitas"[color=red,label=" Pai de "];
"Rosalina Alves" -> "Amélia de Freitas"[color=red,label=" Mãe de "];
"Amélia de Freitas" -> "António Pinto de Lemos"[color=blue,label=" Cônjuge"];
```

Figura 2.13: Excerto do ficheiro *arvore.dot*

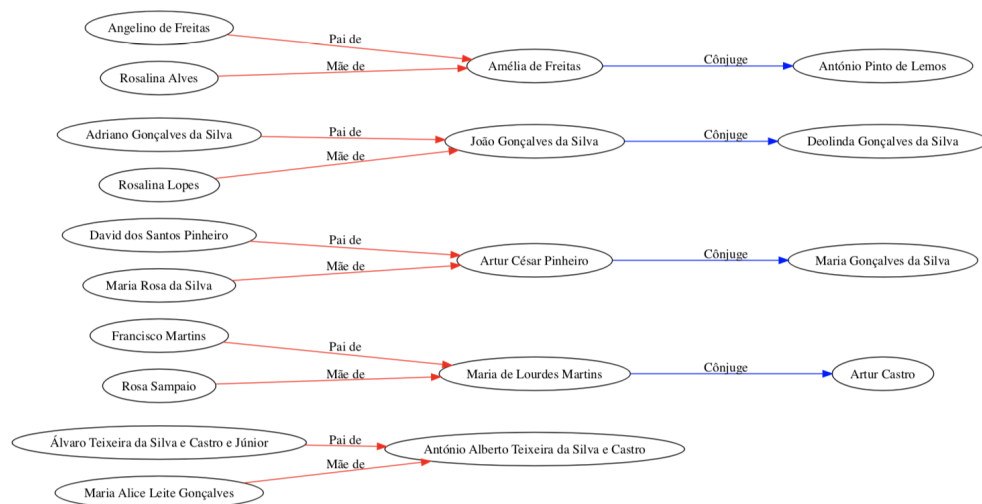


Figura 2.14: Excerto do ficheiro *arcore.pdf*

3. *Apreciação Crítica*

Durante a realização deste segundo trabalho prático desta unidade curricular várias foram as etapas realizadas de forma a obtermos um resultado final fidedigno e consistente.

Apesar de ter sido um trabalho relativamente simples no que toca a identificar campos e respetivos separadores e valores, este foi muito importante para podermos pôr em prática e melhorar a capacidade de análise de ficheiros para obter a informação considerada útil para a resolução de um problema. Para tal também foi necessário entendermos o processo de criação assim como a sintaxe de **GAWK**.

Tendo em conta as metas deste trabalho e os objetivos propostos no enunciado, achamos que obtivemos um resultado bastante positivo e satisfatório. Estamos também bastante satisfeitos e motivados para os desafios que no futuro esta Unidade Curricular irá nos apresentar pois pudemos aperfeiçoar os conhecimentos antes obtidos ao longo da realização deste trabalho.