

Processamento de Linguagens - TP1a

Carlos Pereira (A61887)

João Barreira (A73831)

Rafael Costa (A61799)

Março 2017

Índice

1	Introdução	4
2	Processador de transações da Via Verde	4
2.1	Número total de 'entradas' em cada dia do mês	4
2.1.1	Expressões Regulares e Ações Semânticas	4
2.1.2	Estrutura de Dados Globais	4
2.1.3	Filtro de Texto	5
2.2	Lista dos locais de 'saída'	5
2.2.1	Expressões Regulares e Ações Semânticas	5
2.2.2	Estrutura de Dados Globais	5
2.2.3	Filtro de Texto	6
2.3	Total gasto no mês (total e apenas em 'parques')	6
2.3.1	Expressões Regulares e Ações Semânticas	6
2.3.2	Estruturas de Dados Globais	6
2.3.3	Filtro de Texto	6
3	Album fotográfico em HTML	7
3.1	Página <i>HTML</i> com comandos <i>IMG</i>	7
3.1.1	Expressões Regulares e Ações Semânticas	8
3.1.2	Estruturas de Dados Globais	8
3.1.3	Filtro de Texto	8
3.2	Página <i>HTML</i> com âncoras	9
3.2.1	Expressões Regulares e Ações Semânticas	9
3.2.2	Estruturas de Dados Globais	10
3.3	Filtro de Texto	10
4	Autores musicais	11
4.1	Total de <i>cantores</i> e a lista com os nomes	11
4.1.1	Expressões Regulares	12
4.1.2	Ações Semânticas	12
4.1.3	Estruturas de Dados Globais	12
4.1.4	Filtro de Texto	12
4.2	Todas as canções do mesmo <i>autor</i>	13
4.2.1	Expressões Regulares	13
4.2.2	Ações Semânticas	13
4.2.3	Estruturas de Dados Globais	13
4.2.4	Filtro de Texto	13
4.3	Escrever o nome de cada <i>autor</i> seguido do título das suas canções	14
4.3.1	Expressões Regulares	14
4.3.2	Ações Semânticas	14
4.3.3	Estruturas de Dados Globais	15
4.3.4	Filtro de Texto	15

5	Dicionauro	16
5.1	Criação de uma página <i>HTML</i> com todos os termos e suas respectivas categorias e definições	16
5.1.1	Expressões Regulares e Acções Semânticas	17
5.1.2	Estruturas de Dados Globais	17
5.1.3	Filtro de Texto	18
5.2	Listagem de todos os diferentes domínios e respetivo número de entradas	19
5.2.1	Expressões Regulares e Acções Semânticas	19
5.2.2	Estruturas de Dados Globais	19
5.2.3	Filtro de Texto	19
6	Conclusão	20

1 Introdução

O presente trabalho consiste no desenvolvimento de filtros de texto recorrendo à ferramenta *GAWK*. Os diferentes filtros devem ser produzidos com o recurso a *Expressões Regulares* para detetar *padrões de frases*. Para aumentar estas capacidades, foram propostos quatro exercícios distintos. Neste trabalho, optou-se por resolver todos estes exercícios.

Ao longo deste relatório explicaremos, com detalhe, a resolução de cada exercício proposto. Para cada um é dada ênfase às *Expressões Regulares* e *Ações Semânticas*, bem como eventuais estruturas e variáveis auxiliares utilizadas. Cada secção é também acompanhada pelo código completo do filtro de texto e de um ou mais exemplos da sua execução.

2 Processador de transações da Via Verde

O ficheiro *viaverde.xml* contém o extrato mensal de um cliente do serviço da Via Verde, fazendo referência à data, aos locais de entrada e saída (e.g. estrada ou parque), ao valor pago e à matrícula do veículo, entre outros.

De acordo com estas informações, foi desenvolvido um processador de texto com o auxílio do *GAWK* para ler o ficheiro *XML* e extrair as meta-informações pedidas.

2.1 Número total de 'entradas' em cada dia do mês

Foi feito um processador que lê o ficheiro *XML* e apresenta o número total de entradas para cada um dos dias em cada mês.

2.1.1 Expressões Regulares e Ações Semânticas

- "[>-]" - Esta expressão é usada como **field separator**. Desta maneira, torna-se bastante simples extrair do ficheiro *XML* atributos de cada *tag*, bem como separar os campos de cada uma das datas (mês e dia, nest caso).
- `/<data_entrada>[0-9]/` - Identifica todas as linhas com informação relativa à data de entrada. Através da utilização do *field separator* em questão, é-nos agora possível aceder ao dia e ao mês das datas de entrada (que estão nos campos 2 e 3, respetivamente).

2.1.2 Estrutura de Dados Globais

Neste exercício apenas utilizámos um array bidimensional para guardar o mês e o dia de cada uma das entradas registadas no ficheiro *XML*, bem como duas variáveis auxiliares *auxDay* e *auxMonth* que servem apenas para fazer com que o valor retirado dos campos 2 e 3 seja um inteiro (e não uma string).

2.1.3 Filtro de Texto

```
BEGIN {
    IGNORECASE = 1;
    FS = "[>-]";
}
/<data_entrada>[0-9]/ {
    auxDay = 0 + $2;
    auxMonth = 0 + $3;
    count[auxMonth][auxDay]++;
}
END {
    for (i = 1; i <= 12; i++) {
        if (i in count) {
```

```

        print "M s: " i;

        for (j = 1; j <= 31; j++) {
            if (j in count[i]) {
                print "\tDia: " j " - " count[i][j] " entradas.";
            }
        }
    }
}

```

2.1.4 Exemplos de Execução



```

jpb@linuxmint ~/PL/pl-1617/Via verde $ gawk -f entradas.gawk viaverde.xml
Mês: 7
    Dia: 26 - 3 entradas.
    Dia: 29 - 2 entradas.
    Dia: 30 - 3 entradas.
    Dia: 31 - 1 entradas.
Mês: 8
    Dia: 6 - 4 entradas.
    Dia: 10 - 7 entradas.
    Dia: 11 - 2 entradas.
    Dia: 13 - 5 entradas.
    Dia: 17 - 4 entradas.
    Dia: 18 - 2 entradas.
    Dia: 21 - 4 entradas.

```

Figura 1 - Resultado do filtro de texto *entradas.gawk*

2.2 Lista dos locais de 'saída'

Fez-se um processador que lê o ficheiro *XML* e apresenta a lista dos locais de saída sem repetições e por ordem alfabética.

2.2.1 Expressões Regulares e Ações Semânticas

- "[<>]" - Esta expressão é usada como **field separator**. Desta maneira, torna-se bastante simples extrair do ficheiro *XML* atributos de cada *tag*.
- "/<saída>/ !(\$3 in list)" - Identifica todas as linhas com informação relativa à 'saída' e cujo local de saída (dado pelo campo 3) ainda não esteja no array 'list', que, no fim da execução, conterá o conjunto com os nomes de todos os locais de saída.

2.2.2 Estrutura de Dados Globais

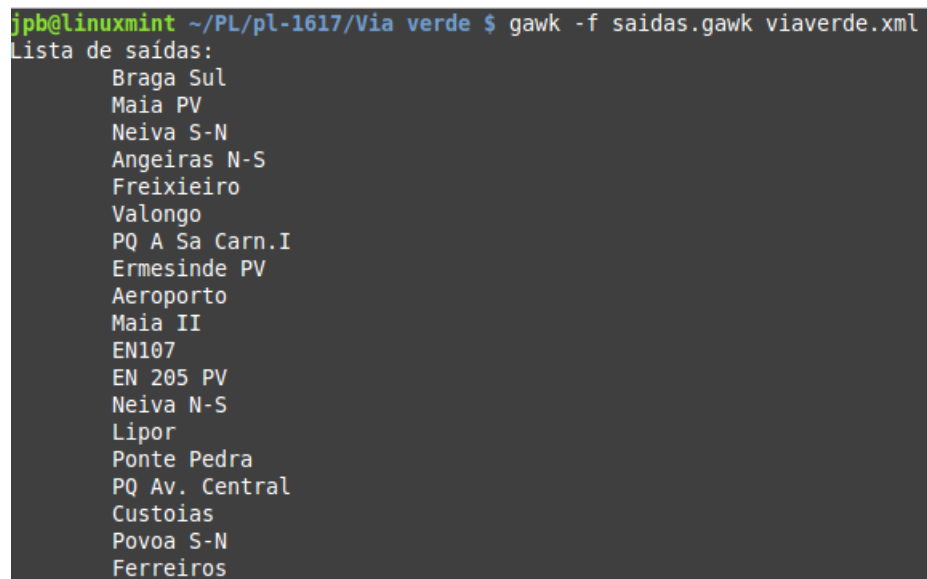
Para este exercício apenas utilizámos um *array* que contém o nome de todos os locais de saída (ordenado alfabeticamente e sem repetições) presentes no ficheiro *XML*.

2.2.3 Filtro de Texto

```
BEGIN {
    IGNORECASE = 1;
    FS = "[><]";
}
/<saida>/ && !($3 in list) {
    list[$3] = $3;
}
END {
    print "Lista de saidas:"

    for (e in list) {
        print "\t" e;
    }
}
```

2.2.4 Exemplos de Execução



```
jpb@linuxmint ~/PL/pl-1617/Via verde $ gawk -f saidas.gawk viaverde.xml
Lista de saidas:
    Braga Sul
    Maia PV
    Neiva S-N
    Angeiras N-S
    Freixieiro
    Valongo
    PQ A Sa Carn.I
    Ermesinde PV
    Aeroporto
    Maia II
    EN107
    EN 205 PV
    Neiva N-S
    Lipor
    Ponte Pedra
    PQ Av. Central
    Custoias
    Povia S-N
    Ferreiros
```

Figura 2 - Resultado do filtro de texto *saidas.gawk*

2.3 Total gasto no mês (total e apenas em 'parques')

Para responder às últimas duas alíneas deste exercício, foi desenvolvido um processador que lê o ficheiro *XML* e apresenta os valores mensais do total gasto e o total gasto apenas em parques.

2.3.1 Expressões Regulares e Ações Semânticas

- "[<>]" - Esta expressão é usada como **field separator**. Desta maneira, torna-se bastante simples extrair do ficheiro *XML* atributos de cada *tag*.
- "/<importancia>/ - Identifica todas as linhas com informação relativa ao valor pago (i.e. importância).
- "/<valor_desconto>/ - Identifica todas as linhas com informação relativa ao desconto. O valor final será calculado subtraindo à importância o valor do desconto.
- "/<tipo>parque/ - Identifica todas as linhas com informação relativa ao 'tipo', selecionando apenas aquelas que são do tipo 'parque'.

2.3.2 Estruturas de Dados Globais

Foram utilizadas as variáveis *imp*, *desc*, *total* e *totalP* para guardar, respetivamente, a importância, o desconto, o gasto total e o gasto em 'parques'.

2.3.3 Filtro de Texto

```
BEGIN {
    IGNORECASE = 1;
    FS = "[><]";
}
/<importancia>/ {
    sub(",", ".", $3);
    imp = $3;
}
/<valor_desconto>/ {
    sub(",", ".", $3);
    desc = $3;
    total += imp - desc;
}
/<tipo>parque/ {
    totalP += imp - desc;
}
END {
    print "Total gasto: " total;
    print "Total gasto em parques: " totalP;
}
```


2.3.4 Exemplos de Execução

```
jpb@linuxmint ~/PL/pl-1617/Via verde $ gawk -f totalGasto.gawk viaverde.xml  
Total gasto: 77.4  
Total gasto em parques: 6.35
```

Figura 3 - Resultado do filtro de texto *totalGasto.gawk*

3 Album fotográfico em HTML

O principal objetivo deste exercício consiste em extrair toda a meta-informação de um ficheiro, em formato *XML*, e gerar uma página *HTML* com essa meta-informação. Esse ficheiro, chamado *lengenda.xml*, possui informação acerca de um conjunto de ficheiros de fotografias e, para cada uma destas, o nome das pessoas envolvidas e o local onde foram tiradas. A página *HTML* deve apresentar todas as fotos listadas no ficheiro *XML*, bem como as pessoas envolvidas em cada uma. Para além disso, deve ser gerada a lista de todos os locais envolvidos, sem repetições. De referir que os ficheiros das fotografias encontram-se em <http://nmp.epl.di.uminho.pt/images/>.

Optou-se por gerar a página *HTML* de duas maneiras distintas: através do comando *IMG* e através de uma âncora com o hiper-link que associe o nome do fotografado ao ficheiro. Apresentam-se, nas secções seguintes, os detalhes de cada uma destas implementações.

3.1 Página *HTML* com comandos *IMG*

Esta implementação gera uma página *HTML* com o seguinte formato:

```
<LI><b>NomeDaPessoa</b></LI>
  <center></center>
```

3.1.1 Expressões Regulares e Ações Semânticas

- "[<>/]" - Esta expressão é usada como *FIELD SEPARATOR*. Desta maneira, torna-se bastante simples extrair do ficheiro *XML* atributos de cada *tag*, bem como nomes de ficheiros entre aspas.
- `/<foto /` - Identifica todas as linhas com informação relativa a uma foto. Como se pode verificar, na definição do *FIELD SEPARATOR*, o nome do ficheiro de uma foto é armazenado no campo número três. Basta armazenar-se o valor desse campo numa variável auxiliar.
- `/<quem>/` - A identificação dos sujeitos envolvidos numa foto é armazenada no campo número três. Usa-se uma função auxiliar para limpar todos os espaços em branco tanto no início como fim da descrição. No final, é acrescentada a informação de uma imagem em formato *HTML* a um ficheiro chamado *index.html*.
- `/<onde>/ !($3 = removeSpaces($3)) in locals` - Filtra todas as linhas que possuam informações do local onde foi tirada uma foto. Uma linha é selecionada se contiver informação de um local que ainda não tenha sido armazenado. Caso estas condições se verifiquem, o local (presente no campo número três) é guardado num *array*.

3.1.2 Estruturas de Dados Globais

As estruturas globais envolvidas neste exercício são as seguintes: um *array locals* para guardar todos os locais (sem repetições) onde foram tiradas as fotos e quatro variáveis globais. Três das variáveis (*fmtLI*, *fmtI* e *end*) possuem valores constantes relativos a código *HTML*. A quarta variável (*image*) serve para guardar o ficheiro da última foto lida.

3.1.3 Filtro de Texto

```
BEGIN {
    IGNORECASE = 1;
    fmtLI = "<li><b>%s</b></li>\n";
    fmtI = "<center><img src=
        \"http://npmp.epl.di.uminho.pt/images/%s\"/>
        </center>\n";
    FS = "[<>\"]";
    end = "</body></html>";
    print "<html><head><meta charset='UTF-8'>
        </head><body>" > "index.html";
}

/<foto / {
    file = $3;
}

/<quem>/ {
    $3 = removeSpaces($3);

    printf(fmtLI, $3) > "index.html";
    printf(fmtI, file) > "index.html";
}

/<onde>/ && !(($3 = removeSpaces($3)) in locals) {
    locals[$3] = 0;
}

END {
    print "<p></p>\n<b>Locais:</b>\n" > "index.html";

    for (i in locals) {
        printf("<li>%s</li>\n", i) > "index.html";
    }

    print end > "index.html";
}
```

```
function removeSpaces(str) {
    sub("^ +", "", str);
    sub(" +$", "", str);

    return str;
}
```

3.1.4 Exemplos de Execução

Apresentam-se de seguida, parte do ficheiro *legenda.xml* (**Figura 4**) e o respetivo resultado numa página *HTML* (**Figura 5**).

```
<foto ficheiro="022-F-16.jpg">
  <quem>Rosa Maria de Oliveira Chaminé Machado</quem>
  <onde>Casa Machado</onde>
  <quando data="2000-09-12"/>
  <facto>O interior da taberna Casa Machado e alguns clientes, na sua maioria
    pescadores.</facto>
</foto>
<foto ficheiro="022-F-17.gif">
  <onde> Igreja Santa Marinha Afurada</onde>
  <quando data="1961-01-15"/>
  <quem> Ana de Lourdes Oliveira Chaminé e António Oliveira Machado</quem>
  <facto>Casamento de António Oliveira Machado com Ana de Lourdes Oliveira Chaminé</facto>
  <legenda> António Oliveira Machado e Ana de Lourdes Oliveira Chaminé, em frente
    ao altar de Nossa Senhora de Fátima, na Igreja de Santa Marinha Afurada, no dia do casamento.</legenda>
</foto>
<foto ficheiro="022-F-15.jpg">
  <onde>Casa Machado, Rua 27 de Fevereiro, Afurada </onde>
  <quando data="2000-09-12"/>
  <quem>Ana de Lourdes Oliveira Chaminé e António Oliveira Machado</quem>
</foto>
```

Figura 4 - Parte do ficheiro *legenda.xml* com três fotos distintas



Figura 5 - Porção da página *HTML* resultante

Todos os diferentes locais são listados no final da página *HTML* de acordo com a seguinte figura:

Locais:

- Na praia de Nazaré
- Igreja Santa Marinha Afurada
- Monte de Santa Luzia, Viana do Castelo
- Casa Machado, Rua 27 de Fevereiro, Afurada
- Casa Machado
- Taberna Fausto, Afurada
- Casa Machado, Afurada, Vila Nova de Gaia
- Taberna do Fausto, Afurada
- Colégio Nossa Senhora de Lourdes, Porto
- Taberna do Fausto, na Afurada
- Casa Machado, Afurada
- Taberna Neca Chaminé, Afurada

Figura 6 - Lista de todos os locais onde foram tiradas as fotos

3.2 Página *HTML* com âncoras

Esta implementação é muito semelhante à implementação com o comando *IMG* mas, acrescenta uma nova funcionalidade. O conjunto de pessoas envolvidas no album é apresentado sem repetições. Cada pessoa tem a si associada uma página *HTML* que dispõe todos as fotos em que está presente. De seguida faremos referência apenas às expressões regulares e ações semânticas que divergem da implementação com *IMG*.

3.2.1 Expressões Regulares e Ações Semânticas

A única expressão regular que é distinta das expressões regulares da outra implementação é a seguinte: `/<quem>/ (length($3) <200)`. Esta expressão filtra todas as linhas com a *tag quem* em que o nome da pessoa envolvida não ultrapasse os duzentos caracteres (de modo a evitar erros ao guardar ficheiros com um nome demasiado extenso).

3.2.2 Estruturas de Dados Globais

Tal como na implementação com *IMG* são utilizadas um conjunto de variáveis constantes que contêm código *HTML*, uma variável para guardar o nome do ficheiro de uma foto e um *array* de locais. Para além destas estruturas é utilizado um *array* de duas dimensões *album[quem][images]*, que associa a uma pessoa um conjunto de fotos onde está presente.

3.3 Filtro de Texto

```
BEGIN {
    IGNORECASE = 1;
    enc = "<html> <head> <meta charset='UTF-8' />
        </head> <body>"
    fmtHREF = "<p><a href=\"%s.html\"> %s </a></p>\n";
    fmtIMG = "<li><center><img
        src=\"%http://npmp.epl.di.uminho.pt/images/%s\" />
        </center></li>\n"
    FS = "[<>\"]";
    end = "</body></html>";
    print enc > "index.html";
}

/<foto / {
    image = $3;
}

/<quem>/ && (length($3) < 200) {
    $3 = removeSpaces($3);
    array[$3][image] = 0;
}

/<onde>/ && !(($3 = removeSpaces($3)) in locals) {
    locals[$3] = 0;
}

END {
    for (i in array) {
        aux = removeInvalidChars(i);

        printf(fmtHREF, aux, i) > "index.html";

        print enc > aux".html";

        for (j in array[i]) {
            printf(fmtIMG, j) > aux".html";
        }

        print end > aux".html";
    }

    print "<p></p>\n<b>Locais:</b>\n" > "index.html";
}
```

```

    for (x in locals) {
        printf("<li>%s</li>\n", x) > "index.html";
    }

    print end > "index.html";
}

function removeSpaces(str) {
    sub("^ +", "", str);
    sub(" +$", "", str);

    return str;
}

function removeInvalidChars(str) {
    sub("[:\t]", "", str);
    return str;
}

```

3.3.1 Exemplos de Execução

Aproveitemos o exemplo fornecido no exercício anterior (**Figura 4**). O resultado é o seguinte:

[António Oliveira Macahdo](#)
[Ana de Lourdes Oliveira Chaminé e António Oliveira Machado](#)
[Manuel de Oliveira Chaminé](#)
[São José, Nossa Senhora de Fátima e Sagrado Coração de Jesus](#)
[Maria Celeste de Oliveira Pereira](#)
[Ana de Loudes Oliveira Chaminé e António Oliveira Machado](#)
[Ana de Lourdes Oliveira Chaminé](#)

Figura 7 - Porção da página *HTML* resultante

A listagem com os locais onde foram tiradas as fotos aparece no fundo da página *HTML*, como no exercício anterior (**Figura 6**).

4 Autores musicais

Existe uma diretoria com vários ficheiros de extensão *.lyr*, que contêm a letra de canções precedida de dois ou mais campos de meta-informação (um por linha). Esta informação pode ser: título da canção, autor da letra, cantor, etc. A letra da música e a meta-informação estão separadas por uma linha em branco.

Foi feito um Processador de Texto com o auxílio do *GAWK* para ler todos os ficheiros *.lyr* e obter informações sobre os cantores, autores e títulos de canções.

4.1 Total de *cantores* e a lista com os nomes

Aqui foi feito um processador que recebe ficheiros de letra de música como *input* e escreve o total e os nomes de todos os cantores.

4.1.1 Expressões Regulares e ações semânticas

- `"*[:;,]"` - Esta expressão é usada como *FIELD SEPARATOR*.
- `/<singer:/` - Identifica todas as linhas com informação relativa a um cantor. Na definição do *FIELD SEPARATOR*, os nomes dos cantores de uma música começam a partir do campo número dois. Após feita a remoção de caracteres desnecessários, os valores desses campos são guardados num *array* em que o índice é o nome do cantor. O contador do número de cantores é incrementado.

No fim imprimem-se os nomes de todos os cantores e o total.

4.1.2 Estruturas de Dados Globais

Foram utilizadas duas estruturas de dados globais: um *array singers*, em que o índice são os nomes de todos os cantores e uma variável *total* que é utilizada como contador de cantores.

4.1.3 Filtro de Texto

```
BEGIN {
    FS = " *[:;,] *"
}

/singer:/ {
    for (i = 2; i <= NF; i++) {
        sub("[ ?()]+$", "", $i);

        if (!($i in singers) && ($i != "")) {
            count++;
        }
    }
}
```



```

        singers[$i] = $i;
    }
}

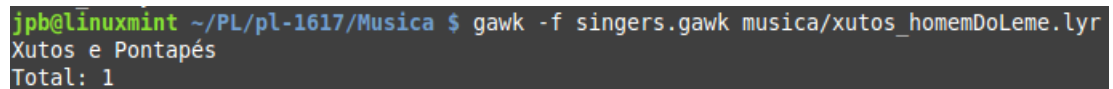
END {
    n = asort(singers);

    for (i = 1; i <= n; i++) {
        print singers[i];
    }

    print "Total: " count;
}

```

4.1.4 Exemplos de Execução

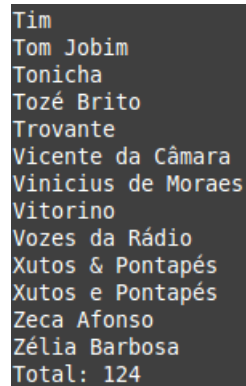


```

jpb@linuxmint ~/PL/pl-1617/Musica $ gawk -f singers.gawk musica/xutos_homemDoLeme.lyr
Xutos e Pontapés
Total: 1

```

Figura 8 - Resultado da execução do filtro de texto aplicado ao ficheiro *xutos_homemDoLeme.lyr*



```

Tim
Tom Jobim
Tonicha
Tozé Brito
Trovante
Vicente da Câmara
Vinicius de Moraes
Vitorino
Vozes da Rádio
Xutos & Pontapés
Xutos e Pontapés
Zeca Afonso
Zélia Barbosa
Total: 124

```

Figura 9 - Parte do resultado do filtro de texto aplicado a todos os ficheiro *.lyr*

4.2 Todas as canções do mesmo *autor*

Fez-se um processador que recebe ficheiros de letra de música como *input* e calcula o número de canções de cada autor.

4.2.1 Expressões Regulares e Ações Semânticas

- `”*[:;,]”` - Esta expressão é usada como *FIELD SEPARATOR*.
- `/<author:/` - Identifica todas as linhas com informação relativa a um autor. Na definição do *FIELD SEPARATOR*, os nomes dos autores de uma música começam a partir do campo número dois. Após feita a remoção de caracteres desnecessários, os valores desses campos são guardados num *array* em que o índice é o nome do cantor e o valor correspondente a esse índice é o número de músicas desse autor. Este contador também é incrementado.

No fim, imprimem-se os resultados da seguinte forma: nome do autor seguido do número de músicas associadas a ele.

4.2.2 Estruturas de Dados Globais

Temos como estrutura global um *array authors*, em que o índice é o nome de um autor e o valor correspondente ao número de músicas associado a esse autor.

4.2.3 Filtro de Texto

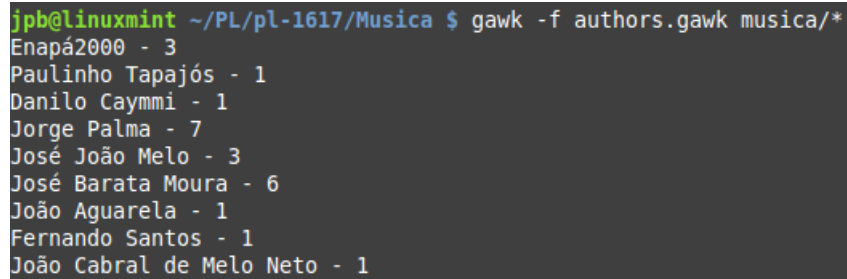
```
BEGIN {
    FS = " *[:;,] *"
}

/author:/ {
    for (i = 2; i <= NF; i++) {
        sub("[ ?()\\t]+$", "", $i);

        if ($i != "") {
            songs[$i]++;
        }
        else {
            songs["Autor desconhecido"]++;
        }
    }
}

END {
    for (i in songs) {
        print i " - " songs[i];
    }
}
```

4.2.4 Exemplos de Execução



```
jpb@linuxmint ~/PL/pl-1617/Musica $ gawk -f authors.gawk musica/*
Enapá2000 - 3
Paulinho Tapajós - 1
Danilo Caymmi - 1
Jorge Palma - 7
José João Melo - 3
José Barata Moura - 6
João Aquarela - 1
Fernando Santos - 1
João Cabral de Melo Neto - 1
```

Figura 10 - Resultado da execução do filtro de texto *authors.gawk*

4.3 Escrever o nome de cada *autor* seguido do título das suas canções

O processador recebe ficheiros de letra de música como *input* e escreve, para cada autor, o título das suas canções.

4.3.1 Expressões Regulares e Ações Semânticas

- `"*[,;,:]"` - Esta expressão é usada como *FIELD SEPARATOR*.
- `/title :/` - As informações relativas aos títulos das canções começam por esta expressão. Quando isto se verificar, o título é guardado numa variável global.
- `/author:` - Os autores de uma canção aparecem depois desta expressão, a partir do campo número dois. Após a remoção de caracteres desnecessários, o título da canção é adicionado a uma matriz cujo índice de linha é o nome do autor e o índice de coluna o título da canção.

Caso não se saiba quem são os autores, o título da canção é adicionado ao *Autor Desconhecido*.

No fim, para cada uma das linhas da matriz *songs*, imprime-se para cada linha, o seu índice e os conteúdos de cada uma das suas colunas.

4.3.2 Estruturas de Dados Globais

Utilizaram-se duas variáveis globais: a variável *song* que guarda o título de uma canção e uma matriz *songs*. Nesta matriz, as linhas correspondem a nome de autores e as respetivas colunas correspondem aos títulos das canções associadas ao autor dessa linha.

4.3.3 Filtro de Texto

```
BEGIN {
    FS = " *[:;,] *"
}

/title: / {
    song = $2;
    sub("\\(\\?\\)", "", song);
    sub("^[ *=]+" , "", song);
}

/author:/ {
    for (i = 2; i <= NF; i++) {
        sub("[ ?()\\t]+$", "", $i);

        if ($i != "") {
            authors[$i][song] = song;
        }
        else {
            authors["Autor desconhecido"][song] = song;
        }
    }
}

END {
    for (a in authors) {
        printf("%s: ", a);

        flag = 0;

        for (j in authors[a]) {
            if (flag == 0) {
                printf("%s", j);
            }
            else {
                printf(", %s", j);
            }
        }

        flag++;

        printf("\n");
    }
}
```

4.3.4 Exemplos de Execução

```
jpb@linuxmint ~/PL/pl-1617/Musica $ gawk -f songs.gawk musica/*
Enapá2000: Marilu, Menina azul (surf), És cruel
Paulinho Tapajós: Andança
Danilo Caymmi: Andança
Jorge Palma: Dá-me lume, Boletim Meteorológico, Pó-de-arroz, Estrela do Mar, Deixa-me
rir, O lado errado da noite
José João Melo: O rei dos matraquilhos, No cume da serra, Blues on de road to Porto
José Barata Moura: Joana come a papa, Fungágá da bicharada, Com a prenda do padrinho,
Resolveram fazer esta canção, Madalena, Olha a bola Manel
João Agualela: A cabana do Pai Tomás
Fernando Santos: Lisboa à noite
```

Figura 11 - Resultado da execução do filtro de texto *songs.gawk*

5 Dicionauro

Uma diretoria designada por *Dicionauro* contém um conjunto de ficheiros com a extensão *.txt* com inúmeras entradas em Português de termos de um *Thesaurus*. Cada termo é iniciado pela sigla *'PT'* e tem a si associado uma ou mais categorias. Um termo pode possuir um conjunto de definições. Uma definição é uma linha iniciada pela sigla *'Def'*.

Descrevem-se, de seguida, os exercícios propostos que têm como base a diretoria *Dicionauro*.

5.1 Criação de uma página *HTML* com todos os termos e suas respectivas categorias e definições

A estrutura da página *HTML* criada neste exercício é a seguinte:

```
<p></p><b>Termo</b>
<p></p><b>Categorias:</b>
<li>Categoria 1</li>
<li>Categoria 2</li>
.
.
.

<p></p><b>Definicoes:</b>
<li>Definicao 1</li>
<li>Definicao 2</li>
.
.
.
```

Ou seja, na leitura de todos os ficheiros presentes na diretoria *Dicionauro* não se devem apresentar termos repetidos, mas sim acrescentar a um termo que já exista novas categorias e definições.

5.1.1 Expressões Regulares e Ações Semânticas

- "[>!<]*pt(_br)?(_pt)?" - Expressão regular atribuída ao *FIELD SEPARATOR*. Ao inspecionarmos vários ficheiros da diretoria em questão, verificamos que nem todos seguem o mesmo padrão. A maior parte dos ficheiros apresenta as linhas respetivas a um termo iniciadas pela sigla *'PT'*. No entanto, verificamos que um pequeno número de ficheiros possuía a sigla *'PT'* precedida ou sucedida pelo seguinte conjunto de caracteres: *<, >, ! e espaço*. Para além disso, uma pequena minoria apresentava distinções para termos em português e em brasileiro, ou seja, *'PT_PT'* e *'PT_BR'*. A expressão regular descrita permite tratar cada um destes diferentes tipos de termos.

- `/^pt/ || /^[><!]*pt/` - Filtra todos as linhas correspondentes a um termo. Tal como foi, falado acima, existem vários formatos para uma linha com um termo. Todos os espaços em branco (tanto no início como no final) de um termo são extraídos, bem como todos os *tabs* que possua. Depois dessa extração, o termo é guardado numa variável auxiliar.
- `/[<>!]*def/` - Esta expressão permite selecionar todas as linhas que contenham uma definição. Novamente a sintaxe de uma linha que contém uma definição pode ser constituída por um conjunto diferente de caracteres. Como o *FIELD SEPARATOR* foi definido de modo a selecionar todos os termos de um ficheiro, é necessário recorrer-se ao uso da função *split* para se extrair a definição propriamente dita. Basta usar-se esta função para o campo número zero (todo o registo) e fazer-se o *split* pela sequência de caracteres *"def"*.
- `/[<>!]*catgra/` - Seleciona todas as linhas que contêm uma categoria. Tal como para uma definição, é realizado um *split* no campo número zero, mas desta vez pela sequência de caracteres *"catgra"*.

5.1.2 Estruturas de Dados Globais

Este exercício recorre ao uso de cinco variáveis auxiliares, de valor constante, para armazenarem meramente código *HTML*. São utilizadas mais três variáveis (*entry*, *def* e *catg*) para guardarem, respetivamente, os valores de um termo, definição e uma categoria. A estrutura mais relevante para este exercício consiste num *array* de três dimensões *entries[entry][catg][def]*. Cada termo deste *array* tem a si associado um conjunto de categorias. Por sua vez, cada categoria tem a si associada um conjunto de definições. Esta estruturação é útil em casos em que uma nova categoria é definida para um mesmo termo noutra ficheiro. De modo a facilitar a compreensão desta estrutura ilustraremos o seguinte exemplo: suponha-se que existem três ficheiros distintos ("A", "B" e "C"). Ambos os ficheiros possuem o mesmo termo ("*corpo*") e um conjunto de definições distintas para este. Suponha-se, também, que no ficheiro "A" a categoria do termo é "*nl*" (tal como no ficheiro "B") e que no ficheiro "C" não há nenhuma informação relativa à categoria do termo. Sendo assim o *array* possui as seguintes configurações: *entries["corpo"]["nl"]["conjunto de definições do ficheiro "A" e "B"]* e *entries["corpo"][""]["conjunto de definições do ficheiro "C"]*.

5.1.3 Filtro de Texto

```
BEGIN {
  IGNORECASE = 1;
  fmtLI = "<li styl , =\"margin-left:50px\">%s</li>\n";
  fmtE = "<p></p><b style=\"font-size:150%\">%s</b>\n";
  fmtCD = "<p></p><b style=\"margin-left:25px\">%s</b>\n";
  FS = "^ [> !<]*pt(_br)?(_pt)?";
  end = "</body></html>";
```

```

        print "<html><head><meta charset='UTF-8' />
              </head><body>" > "index.html";
    }

/^pt/ || /^[>< !]*pt/ {
    entry = removeWhiteSpaces($2);
}

/[> <!?]*def/ {
    split($0, target, "def");
    def = removeWhiteSpaces(target[2]);
    entries[entry][catg][def] = def;
}

/[> <!?-]*catgra/ {
    split($0, target, "catgra");
    catg = removeWhiteSpaces(target[2]);
}

END {
    for (i in entries) {
        if (length(i) > 1) {
            printf(fmtE, i) > "index.html";

            printf(fmtCD, "Categorias:") > "index.html";

            for (c in entries[i]) {
                if (length(c) > 1) {
                    printf(fmtLI, c) > "index.html";
                }
            }

            printf(fmtCD, "Definiciones:") > "index.html";

            for (j in entries[i]) {
                for (w in entries[i][j]) {
                    if (length(w) > 1) {
                        printf(fmtLI, w) > "index.html";
                    }
                }
            }
        }
    }

    print end > "index.html";
}

```



```

}

function removeWhiteSpaces(str) {
    sub("\\t", "", str);
    sub("^ +", "", str);
    sub(" +$", "", str);

    return str;
}

```

5.1.4 Exemplos de Execução




Como o termo '*costas*' aparece em vários ficheiros, aparecem também várias entradas no ficheiro *HTML* resultante (**Figura 12**).

gua

Categorias:

- nf

Definições:

- a gua  um animal grande e forte que as pessoas usam para passear nele ou para puxar carroas

costas

Categorias:

- substantivo masculino
- nm

Definições:


- estilo em que o nadador se desloca com o corpo recto, girando os braços para trás e batendo os pés
- as costas ficam na parte de trs do tronco.

Figura 12 - Resultado da execução do filtro de texto '*entradas.gawk*'

5.2 Listagem de todos os diferentes domínios e respetivo número de entradas

A última alínea deste exercício corresponde a uma listagem de todos os domínios distintos presentes nos ficheiros da diretoria *Dicionauro*. Para além disso, deve-se indicar para cada domínio a quantidade de termos pertencentes.

5.2.1 Expressões Regulares e Ações Semânticas

- `"^ [>!] *dom +"` - Expressão atribuída ao *FIELD SEPARATOR*. Verificou-se que, em certos ficheiros, uma linha que caracteriza um domínio é iniciada pelo seguinte conjunto de caracteres: *espaço*, *,*, *>*, *!* e *<*. Para além disso, em algumas linhas a palavra *dom* é sucedida por um ou mais espaços.
- `/^ [>!] *dom / length($2) > 0` - Filtra todas as linhas que contenham informação acerca de um domínio não vazio. O domínio é guardado num *array* e, caso este já exista, incrementado o seu número de termos.

5.2.2 Estruturas de Dados Globais

Nesta alínea apenas foi usada a seguinte estrutura de dados: um *array* chamado *dom* em que os seus índices correspondem a diferentes domínios. Em cada posição do *array dom* consta o número de termos correspondentes.

5.2.3 Filtro de Texto

```
BEGIN {
    IGNORECASE = 1;
    FS = "[&> !<]*dom ";
}

/^[&>< !]*dom / && length($2) > 0 {
    dom[norm($2)]++;
}

END {
    for (i in dom) {
        print i ": " dom[i];
    }
}

function norm(str) {
    sub("^ +", "", str);
    sub(" +$", "", str);
    return tolower(str);
}
```

5.2.4 Exemplos de Execução

```
jpb@linuxmint ~/PL/pl-1617/Thesaurus $ gawk -f dominios.gawk Dicionauro/*
animais domsticos: 15
alimento: 55
desporto: 151
vida: 5
actividade: 9
tempo: 43
ciência: 1
anatomia: 1
propriedade: 83
motociclismo: 3
equitação e corrida de cavalos: 8
casa: 22
árvore: 4
medicina: 2
seres vivos: 1
biologia: 1
termos gerais: 26
corpo humano: 67
zoologia: 5
alimentao: 2
anatamia: 3
atletismo: 30
botânica: 142
ferramenta: 1
animal: 38
anatomia: 238
```

Figura 13 - Resultado da execução do filtro de texto '*dominios.gawk*'

6 Conclusão

Com a realização deste trabalho, pudemos pôr em prática os conceitos adquiridos nas aulas teórico-práticas sobre a ferramenta *GAWK*.

Percebemos a importância do uso das expressões regulares e ações semânticas, de forma a podermos pegar num ficheiro, obter as informações que necessitamos e, com elas, chegar a um conjunto de conclusões.

Tomámos consciência de que esta tarefa nem sempre é fácil (devido à complexidade dos textos em questão), o que faz com que a definição das expressões regulares para conseguirmos obter as informações que necessitamos nem sempre seja uma tarefa trivial. Um bom exemplo deste tipo de dificuldade, foi a nossa tentativa de separar as pessoas no ficheiro *xml* do exercício 2 relativo ao álbum de fotografias.

Assim sendo, em retrospectiva, achamos que a realização deste primeiro trabalho prático foi enriquecedora e nos deu uma boa base relativamente ao processamento de ficheiros textuais.