Processamento de Linguagens (3º ano de Curso)

Trabalho Prático 1

Relatório de Desenvolvimento

José Silva	Pedro Cunha	Gonçalo Moreira
(A74601)	(A73958)	(A73591)

12 de Março de 2017

Resumo

Isto é um resumo do relatório de sadsa focando o contexto do trb (muito sucinto), Cenas

Conteúdo

Capítulo 1

Introdução

O avanço tecnológico dos últimos anos trouxe consigo a inevitabilidade de processar cada vez mais texto. Por parte de grande parte dos utilizadores existe a necessidade frequente de fazer mudanças ou extrair determinadas linhas de grandes quantidades de texto onde certos padrões são bastante evidentes. O uso de expressões regulares, que proporcionam um método eficiente, poderoso e flexível no que toca ao processamento de texto, combinado com as ferramentas que a linguagem AWK (linguagem de programação bastante mais fácil de utilizar que as linguagens mais convencionais) proporciona um método eficiente para solucionar as necessidades descritas acima. Das ferramentas descritas anteriormente destacam-se funções capazes de manipular strings e a utilização de arrays associativos, estruturas de dados bastante uteis e que não são disponibilizadas por todas as linguagens de programação.

Neste primeiro trabalho prático da unidade curricular de "Processamento de Linguagens", através dos meios descritos anteriormente, vai desenvolvido um filtro de texto capaz de realizar o processamento de transações presentes nos extratos mensais disponibilizados pela empresa Via Verde.

Estrutura do Relatório

No capítulo 1 faz-se uma pequena introdução ao problema e às ferramentas utilizadas para a resolução deste. Para além disso, é descrita de uma forma breve a estrutura do relatório.
No capítulo 2 faz-se uma análise breve mas mais detalhada do problema escolhido pelo grupo de trabalho.
No capítulo 3 é descrito de uma forma sumarizada como procedemos para solucionar as várias questões propostas pelo enunciado.

No capítulo 4 são apresentados alguns testes e respectivos resultados para comprovar o respectivo funcionamento da solução apresentada.

Finalmente, no capítulo 5 termina-se o relatório com uma síntese do que foi dito, as conclusões e o trabalho futuro.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

É fornecido um ficheiro xml, que corresponde ao extrato mensal emitido pela Via Verde para um dos seus utentes. Pretende-se que se desenvolva um *Processador de Texto* para ler esse mesmo ficheiro e retirar a informação requisitada, apresentada em mais detalhe a baixo, na Especificação dos Requisitos.

2.2 Especificação dos Requisitos

2.2.1 Dados

Como já foi referido, é fornecido um ficheiro xml com informação correspondente ao estrato mensal emitido pela Via Verde para um dos seus utentes. Este ficheiro contém, no início, informação do utente e também também especifica o mês de emissão. Depois destes dados, seguem-se todas as transações do cliente. Cada uma destas transações, contém a data, as horas de entrada e saída, o local de entrada e saída, a importância paga, o desconto, o iva, o operador e ainda o tipo de transação(Portagens ou Parques de estacionamento). No final do ficheiro, é apresentado o total gasto no mês de emissão e ainda o valor que diz respeito ao iva.

2.2.2 Pedidos

O utente pode ter efetuado entrada em alguns dos dias do mês a que este extrato se refere. Posto isto, é solicitado que se calcule o número de entradas em cada dia do mês. Considera-se também relevante obter a lista de locais de saída pelos quais o utente passou durante todo o mês. Chegando ao fim do mês, uma das informações mais importantes para o utente é, muito provavelmente, o total gasto nesse mesmo mês. É então pedido que se obtenha o total que o utente gastou. Pretende-se também que seja calculado quanto desse total é que corresponde ao valor despendido apenas em parques de estacionamento.

Para obter a informação necessária será utilizado o Sistema de Produção GAWK, especificando os padrões de frases que se pretende encontrar com recurso a expressões regulares.

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

Pensando nos requisitos para este projeto, é fácil de perceber que será necessário guardar informação relativa a alguns tópicos. Por exemplo, será crucial guardar a informação que diz respeito à quantidade de entradas num determinado dia. Para guardar informação deste tipo, serão utilizados arrays associativos. Embora apenas seja requisitado que se liste os diferentes locais de saída, serão utilizados arrays associativos de forma a guardar também quantas vezes passou por cada um desses locais. Relativamente a cada dia, serão ainda guardados os totais gastos em cada dia num array associativo que, embora não seja um requisito, se considera informação importante. Acrescentando ainda aos extras já apresentados, armazena-se informação relativa ao valor despendido em parques e também em portagens. Informação como o total gasto no mês não precisa de ser guardada em nenhuma estrutura de dados, bastando usar um float para o efeito.

3.2 Algoritmos

No início do programa, serão atribuidos todos os valores necessários a algumas variáveis que serão uteis para o corpo do programa. Por exemplo strings, inteiros e formatos para a função printf. É também muito importante a definição de um *Field Seperator* adequado para facilitar o processamento do ficheiro.

O corpo do programa será na forma de condição e ação respetiva, como é típico usando GAWK. Na maioria das condições, estão especificadas expressões regulares que um ou mais 'fields' da linha atual tem que satisfazer. Sabendo que esse 'field' satisfez a expressão regular, guarda-se informação valiosa para o resultado final do programa.

Chegando à fase final de execução trata-se a informação obtida, de forma a apresentar os resultados em formato apelativo. Por exemplo, usando html com strings definidas no início e não só. Existirá, muito provavelmente, um ciclo para quase todos os arrays associativos de forma a apresentar a informação armazenada no mesmo. Um destes ciclos, deverá juntar a quantidade de entradas num dia e o total gasto no mesmo. Depois de tratar toda a informação, no caso de ser usado html, coloca-se o final no ficheiro fechando o body e o html.

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

Uma das decisões tomadas logo no início da implementação foi usar html como tipo de ficheiro de 'output'. Este ficheiro utiliza estilos escritos em css. Também é utilizada a font-awesome para juntar icons que refletem o conteúdo de cada linha apresentada. O objetivo do uso destes recursos é tornar o conteúdo do ficheiro apelativo. Uma alternativa a esta abordagem seria utilizar o standard output para apresentar os resultados, mas considerou-se que o html 'ganhava' em tudo, excepto na complexidade de construção. Tendo apenas esse problema, que não causa muito transtorno, achou-se por bem optar pelo html. Como se pode verificar no bloco begin(linhas 3 a 24), foram criados vários formatos e strings para auxiliar na criação do ficheiro.

Das decisões tomadas, a mais importante foi, talvez, decidir qual *Field Seperator* usar. Depois de observar bem o ficheiro xml, chegou-se à conclusão que seria oportuno utilizar como *Field Seperator* a expressão regular na linha 4 do código: [<>]. Assim, é possível obter o que esta dentro de cada elemento xml apenas acedendo à posição 3. Isto facilita muito a obtenção de informação. Com outra abordagem, podia ter sido usado o *Field Seperator* default, mas dessa forma seria necessário um tratamento muito mais complexo das strings em cada posição para retirar dados importantes.

Durante a implementação do programa, não houve nenhum problema que mereça destaque. Os algoritmos utilizados são relativamente simples e as expressões regulares também. A expressão mais 'complexa' utilizada, foi a que garante que a data de entrada seja no formato correto(removendo ocorrências de null). Esta expressão está presente na linha 57 do código.

4.2 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos (valores introduzidos) e os respectivos resultados obtidos:

Capítulo 5

Conclusão

Síntese do Documento .
Estado final do projecto; Análise crítica dos resultados .
Trabalho futuro.

Apêndice A

Código do Programa

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

procViaverde

```
#!/usr/bin/gawk -f

BEGIN {
    FS = "[<>]";
    file = "index.html";
    icon = "<i class='fa fa-calendar'>"
    fmtEntrad = "<li>"icon" <b>Dia %s:</b> %d (%.2f€)</i></li>\n";
    fmtTotais = "<li>Total gasto apenas em %s: %.2f€</li>\n";
    fmtli = "<li> %s</li>\n";
    fmth5 = "<h5 class='bg-3'>%s</h5>\n";
    fmth4 = "<h4 class='bg-2'>%s</h4>\n";
    fmth2 = "<h2>%s</h2></div>\n";
    icon = "<i class='fa fa-road'>";
    ini = "<!DOCTYPE html> <html> <head>\n";
    tit = "<title> Via Verde </title>";
    enc = "<meta charset='utf-8'/> </head> <body>";
    end = "</body> </html>";
    beg = "<div class='bg-4 title'><h1>"icon" Via Verde </i></h1>";
    css = "<link rel='stylesheet' href='css/styles.css'>";
    fta = "<link rel='stylesheet' href='css/fa/css/font-awesome.css'>";
    ref = "<a target='_blank' href='%s'>%s</a>";
    maps = "https://www.google.pt/maps/place/";
    infoCliente = 0;
}

NR == 1 {
    print ini > file;
    print tit > file;
    print css > file;
    print fta > file;
    print enc > file;
    print beg > file;
}

$2~/MES_EMISSAO/{
    s = "Mês Emissão: " $3;
    printf(fmth2, s) > file;
}

```

```
40 $2~/NIF/{
41     infoCliente++;
42     print "<ul>" > file;
43 }
44
45 infoCliente {
46     icon = "<i class='fa fa-user'> ";
47     str = icon"<b>"$2"</b>:" $3"</i>";
48     gsub("_"," ", str);
49     printf(fmtli, str) > file;
50
51     if($2~/CODIGO_POSTAL/) {
52         print "</ul>" > file;
53         infoCliente--;
54     }
55 }
56
57 $2~/DATA_ENTRADA/ && $3~/[0-9-]+/ {
58     dias[$3]++;
59     ultDia = $3;
60 }
61
62 $4~/\/SAIDA/ {
63     locaisSaida[$3]++;
64 }
65
66 $4~/\/ENTRADA/ {
67     locaisEntra[$3]++;
68 }
69
70 $2~/IMPORTANCIA/ {
71     gsub(",",".", $3);
72     totalGasto += $3;
73     ultimoGasto = $3;
74     totDias[ultDia] += $3;
75 }
76
77 $2~/VALOR_DESCONTO/ {
78     gsub(",", ".", $3);
79     totalGasto  -= $3;
80     ultimoGasto -= $3;
81     totDias[ultDia] -= $3;
82 }
83
84 $2~/TIPO/ {
85     totalLocais[$3] += ultimoGasto;
86 }
87
88 END {
89     str = "<h2 class='bg-4'>Informação:</h2>";
90     print str > file;
91
92     str = "N° de entradas em cada dia do mês:";
93     printf(fmth4, str) > file;
94     print "<ul>" > file;
95     for (i in dias)
96         printf(fmtEntrad, i, dias[i], totDias[i]) > file;
97     print "</ul>" > file;
98
```

```
99
100     printf(fmth4, "Locais de saída:") > file;
101     print "<ul>" > file;
102     for (i in locaisSaida) {
103         aux = sprintf(ref, maps i, i);
104         icon = "<i class='fa fa-map-marker'> ";
105         str = icon aux " </i>" "("locaisSaida[i]")";
106         printf(fmtli, str) > file;
107     }
108     print "</ul>" > file;
109
110
111     str = "Total gasto no mês:";
112     printf(fmth4, str) > file;
113     str = "Total:" " " totalGasto"€";
114     printf(fmth5, str) > file;
115     print "<ul>" > file;
116     for(i in totalLocais)
117         printf(fmtTotais, i, totalLocais[i]) > file;
118     print "</ul>" > file;
119
120     print end > file;
121 }
122
123 function duracao(a) {
124
125 }
```