Processamento de Linguagens e Compiladores
(3º ano de LCC - Pedro Rangel Henriques)

Trabalho Prático Nº2 (GAWK)

2.2 Processador de Processos de Formação
Relatório de Desenvolvimento

Francisca Fernandes (A72450)

Francisco Oliveira (A82066)

Maria Luísa Silva (A82124)

25 de Julho de 2021

Resumo

Isto é um resumo do relatório de *Processamento de Linguagens e Compiladores* focado no contexto do trabalho processador de processos de formação, descrição do problema, as decisões e os resultados, tanto como as dificuldades que nos foram aparecendo durante a resolução do mesmo.

Conteúdo

Capítulo 1

Introdução

O trabalho prático 2.2 Processador de Processos de Formação, surge inserido no âmbito da disciplina de *Processamento de Linguagens e Compiladores*. É o segundo trabalho prático desta unidade curricular, abordando o uso da ferramenta **GAWK**.

A linguagem de programação **GAWK** é baseada na linguagem C, e é utilizada frequentemente por desenvolvedores para processar textos e manipular arquivos.

Tem como paradigmas linguagem de script, procedural e orientada a eventos

1.1 Enunciado

Analise com todo o cuidado o ficheiro **natura.di.uminho.pt/ jj/pl-19/TP2/formacao.csv** o qual contém informação detalhada sobre a criação e gestão de cursos de formação profissional.

Construa então um ou mais programas **GAWK** que processem esse arquivo de modo a:

a) limpar o ficheiro dado (criar um segundo ficheiro pre-processado) retirando linhas vazias extra e linhas cujos campos estão todos vazios; sempre que 'Estado' esteja vazio, esse campo deve tomar valor 'NIL'. Este ficheiro pre-processado deve ser usado nas alíneas seguintes!

b) contar o número de registos que apresentam um código numérico e mostrar para esses, num formato legível, o dito código da ação de formação, o seu títtulo, descrição e notas.

c) identificar os tipos diferentes e calcular o número de processos por tipo.

d) desenhar um grafo (em DOT) que relacione cada ação de formação (identificada pelo seu código) com todos os Diplomas jurídico-administrativos usados.

1.2 Descrição do Problema

Este trabalho consiste em processar um ficheiro de texto utilizando a ferramenta **GAWK** de forma a que o output seja o especificado no enunciado.

Capítulo 2

Codificação e Testes

2.1 Decisões e Problemas de Implementação

2.1.1 Decisões

Para a formatação do texto decidmos retirar linhas extra vazias , linhas cujos campos estivessem todos vazios e colocar 'NIL' se o primero campo fosse vazio, tal como o enunciado pedia, mas também remover todos os ";"que apareciam a mais no final de cada linha.

Para a alínea b) decidimos considerar um formato numérico como uma sequência de digitos com o possível separador '.' entre digitos. Usamos também uma variável *count* para funcionar como contador de número de códigos já vistos, e o array *seen* para garantir que cada código só é contabilizado e imprimido uma unica vez.

Para contabilizar o número de processos por tipo decidimos usar um array como dicionário em que as chaves são os tipos de processo e os valores são o número de processos associado a esse mesmo tipo. Deste modo torna-se fácil imprimir todos os tipos diferentes e o número de processos por cada tipo.

Na alínea d) optamos por criar um nodo por cada ação de formação(cujo texto é o próprio código) e outros dois nodos para os Diplomas jurídico-administrativos com uma aresta para a respetiva ação de formação.

2.1.2 Problemas de Implementação

A dificuldade deste trabalho foi limpar o ficheiro dado, correspondente à alínea a) do nosso enunciado, mais especificamente encontrar um *record separator* que resolvesse o problema.

Ao fim de muita pesquisa e de várias tentativas erros acabamos por conseguir encontrar um que nos deu o resultado pretendido .

A dificuldade por nós encontrada na alínea b) foi, o facto de estarmos na dúvida se era para reconhecer números ou formatos numéricos.

Depois de uma melhor examinação do ficheiro optamos por reconhecer formatos numéricos.

A alínea c) foi a mais acessível portanto não tivemos dificuldades significativas.

A alínea d) foi a alínea que nos causou mais transtorno neste trabalho por não termos experiência com o formato DOT.

2.2 Testes realizados e Resultados

Os testes realizados foram feitos no segundo texto pre-processado que será enviado em anexo .

2.2.1 resultado da alínea a)

O resultado desta alínea é o ficheiro processado, que vai juntamente com o relatório, em anexo.

2.2.2 resultado da alínea b)

Este é o nosso resultado da alínea b), em que contamos o número de registos que apresentam um código numérico e mostramos para esses, o código da ação de formação, o seu títtulo, descrição e notas.

```
Código:  750
Titulo:  PRESTAÇÃO DE SERVIÇOS DE ENSINO E FORMAÇÃO
Descrição:  "Relativo à prestação de serviços no domínio da educação/ensino/qualificação da população,
independentemente da idade ou do contexto (escolar, académico, profissional ou outro) - inclui a educação
pré-escolar, o ensino básico e secundário, o ensino superior, a educação extraescolar e todos os cursos
de formação, de qualificação profissional e valorização permanente, em qualquer área do conhecimento. "
Notas:  "Inclui as ações de formação para a qualificação dos recursos humanos da administração pública, bem
como a realização de estágios.# Aplicável tanto às entidades que prestam os serviços referidos como às
que solicitam a prestação dos mesmos.#"

...

Número total de códigos: 29
```

2.2.3 resultado da alínea c)

A seguir são apresentados os tipos diferentes de processos e número de processos por tipo, assim como é pedido pela alínea c).

```
PC:12
PE:5
```

2.2.4 resultado da alínea d)

Para o nosso resultado da alínea d) decidimos colocar um print de uma parte do nosso grafo onde seja evidente as relações de cada ação de formação com todos os Diplomas jurídico-administrativos usados, sendo estes os diploma complementar e diploma REF.

Figura 2.1: Resultado alínea d)

Em anexo enviaremos o resultado final de todo o grafo gerado em DOT.

Capítulo 3

Conclusão

O **GAWK** é uma linguagem e processador de padrões em texto.

Evidencia-se, por exemplo, em relação ao **Flex**, a facilidade no tratamento de ficheiros de texto com formatação em campos, devido à possiblidade de acessar cada campo individualmente.

A função básica do **GAWK** é processar linha a linha num determinado ficheiro de entrada que possua certos padrões especificados no programa. Para cada padrão deve haver uma ação associada, isto é, quando uma linha corresponde a um dos padrões, o **GAWK** realiza a ação correspondente naquela linha, caso contrário, ignora. Depois continua processando as linhas de entrada desta forma até encontrar o fim do ficheiro.

A realização deste trabalho permitiu-nos perceber melhor o funcionamento e a aplicação do **GAWK**. É um excelente filtro e processador de ficheiros de texto, especialmente quando queremos processar uma grande quantidade de informação, sendo mais fácil de usar do que a maioria das linguagens de programação convencionais.

De um modo geral fazemos um balanço positivo do nosso trabalho. Embora na alínea d) tivemos algumas dificuldades a resolve-la, porque nunca tínhamos usado o formato **DOT**, mas após algum estudo conseguimos fazer o que era pedido.

Apêndice A

Código do Programa

Lista-se agora o código da nossa resolução da alínea a) do problema

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

BEGIN {FS = ";"; RS = ";\r\n(\r\n)+";
ORS = "\r\n"
}

NR >= 1 {


gsub("\r\n", " ", $0)
vazio = 1;

# remove pontos e virgulas a mais no final
for(i=NF;i>1; i--)
if ($i =="")
NF=NF-1
else break

#retorna 1 se todos os fields forem vazios e 0 se não
for (i=1; i<=NF; i++)
{
if($i != "" && $i != " ")
{
vazio = 0;
break;
}
}
# coloca nil se o primeiro field for vazio e coloca ";" no final de cada field
if(vazio != 1)
{
if ($1=="")
$1= "NIL"

for (i=1; i<=NF; i++)
$i=$i";"

print $0
```

```
}
}
```

Lista-se agora o código da nossa resolução da alínea b) do problema

```
BEGIN { FS = ";"; count = 0; }

$2 ~ /^[ ]*([0-9]+(\.[0-9]+)*)+[ ]*$/ {
if(!seen[$2]++)
{
count++;
print "Código: " $2;
print "Titulo: " $3;
if($4 != "" && $4 != " ")
{
print "Descrição: " $4;
}

if($5 != "" && $5 != " ")
{
print "Notas: " $5;
}
}
}

END { print "Número total de códigos: " count; }
```

Lista-se agora o código da nossa resolução da alínea c) do problema

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

BEGIN {FS = ";" ;}
NR >1 {
if ($10 != " " && $10!= "")
tipo[$10]++
}
END {
for(i in tipo){
print i ":" tipo [i]
}
}
```

Lista-se agora o código da nossa resolução da alínea d) do problema

```
BEGIN{
FS=";";
dot = "grafo.dot";
    print "digraph{" > dot;
    print "rankdir=LR" > dot;
}
NR>2
{
gsub("\"", "\\\"", $8);
i++;
print "DC" i "[label=" "\"" $9 "\"" "]" > dot;
```

```
print "DR" i "[label=" "\"" $8 "\"" "]" > dot;
print "E" i "[label=" "\"" $2 "\"" "]" > dot;
print "DC" i "->" "E" i "[label=\"Diploma complementar\"]" > dot;
print "DR" i "->" "E" i "[label=\"Diploma REF\"]"> dot;

}
END{
print "}" > dot;
}
```