Processamento de Linguagens e Compiladores
(3º ano de LCC)

Trabalho Prático Nº2 (GAWK)

2.3 Processar as Pessoas listadas nos Róis de Confessados com o Gawk
Relatório de Desenvolvimento

João Ferreira (A76628)

Luís Daniel Félix (A74246)

Maria Manuela Silva (A74408)

14 de Janeiro de 2019

Conteúdo

Capítulo 1

Introdução

O trabalho prático 2.3, Processar as Pessoas listadas nos Róis de Confessados com o Gawk, 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**. Ao longo deste documento podemos encontrar a exposição do problema, a sua análise, a forma como o interpetamos e resolvemos e as dificuldades que surgiram ao longo da sua realização.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Foi-nos fornecido um ficheiro em anexo de nome *"processos.txt"*,que contem processos das pessoas listadas nos Róis de confessados. Pretende-se desenvolver um Processador de Texto com o ***GAWK*** para ler um o ficheiro anexado, ou seja, o objetivo é contruir um ficheiro chamado *expprocessos.gawk* para processar o texto 'processos.txt com o intuito de calcular frequências de alguns elementos (a ideia é utilizar arrays associativos para o efeito).

2.2 Especificação do Requisitos

2.2.1 Dados

Para resolver o problema que nos foi proposto dividimos o ficheiro *expprocessos.gawk* em 3 para que fossem satisfeitos os pedidos. No ficheiro *processos.txt* temos diferentes tipos de dados. No primeiro campo temos o número do processo, no segundo campo tem a data do processo,nos restantes campos encontram-se os nomes das pessoas envolvidas no processo, tipos de realções entre elas, entre outras informações. Para acedermos aos varios campos e fazer os calculos pedidos, tivemos de recorrer a arrays.

2.2.2 Pedidos

Neste trabalho, foram-nos dadas 3 alíneas para resolver. Na alinea (a) é-nos pedido que calculemos a frequência de processos por ano (primeiro elemento da data); Na alinea (b) é-nos pedido que calculemos a frequência de nomes (considera um nome uma palavra e propaga o cálculo por todos os campos que contenham nomes); Na alinea (c) é-nos pedido que calculemos a frequência dos vários tipos de relação: irmão, sobrinho, etc.

Capítulo 3

Conceção/desenho da Resolução

Para a realização das 3 alineas, criamos 3 fcheiros, um para cada uma das alineas denominados de *exepprocessosa.gawk*, *exepprocessosb.gawk*, *exepprocessosc.gawk* respetivamente. Na alinea **a** e **b** vamos buscar o que desejamos aos campos queremos sem o uso de expresões regulares, recorrendo apenas a funções do gawk e da linguagem C, ja na alinea **c** recorremos tambem ao uso de ER's (Expressões Regulares).

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

Depois de estudarmos o enunciado, fizemos em relativamente pouco tempo a as alineas a e c. A nossa dificuldade prendeu-se no problema b uma vez que temos de imprimir todos os nomes e os mesmos do ultimo campo nao imprimem devido ao formato em que está o texto.

4.2 Testes realizados e Resultados

Na **alínea a)**,usamos o *split* para passar o campo 2 (que é o campo que contem a data) para um array chamado *data*, sempre que encontra **-**. Seguidamente, é guardado cada ano (que se encontra na posição 1 do array data, data[1]) no array count e é feita a sua contagem.

```
BEGIN {FS = "::"}

      {
      split($2,data,"-");
      count[data[1]]++;
      }

END {
    for (ano in count) {print "No ano " ano " há " count[ano] " processos"};
    }
```

Figura 4.1: Implementação da alinea a

Na **alinea b)** começamos por dividir os campos e começar a contar a frequência da nomes através de um array denominado *freq*. Seguiu-se um caso particular para o campo 6 onde se encontram campos estruturados de forma diferente. Devido a essa estrutura diferente, fomos tentando ir buscar os nomes existentes sendo que tivemos sucesso em alguns mas noutros não. A codificação da alinea b encontra-se de seguida na figura 4.2.

```
BEGIN{FS = "::"}
{
   for(i=3; i<NF-1; i++){
       {freq[$i]++};
    }
    {split($6,name,",");
    freq[name[1]]++;
    {split(name[1],name3,".")
      freq[name3[3]]++;}
    {split(name[2],name2,".");
      freq[name2[4]]++;}
    }
}

END {
     for (nome in freq) {print "O nome " nome "  aparece " freq[nome] "  vezes"}
     }
```

Figura 4.2: Implementação da alinea b

Na **alinea c)** implementamos um ciclo para correr todas as linhas e campos do programa e porcurar as relações existentes através de expressões regulares, e incrementar de modo a dar o número de relações de cada tipo que de facto existem. A codificação da alinea c encontra-se nas duas imagens a baixo.

```
BEGIN{FS = "."}
{
   for(i=1; i<=NF; i++){
        if($i ~ /,Irmaos/)
              {rela["Irmãos"]++};


        if($i ~ /,Irmao/)
              {rela["Irmao"]++};

        if($i ~ /,Tio Paterno/)
              {rela["Tio Paterno"]++}


        if($i ~ /,Tio Materno/)
             {rela["Tio Materno"]++};


        if($i ~ /,Pai/)
             {rela["Pai"]++};

        if($i ~ /,Pais/)
             {rela["Pais"]++};

        if($i ~ /,Sobrinho Materno/)
             {rela["Sobrinho Materno"]++};

        if($i ~ /,Sobrinho Paterno/)
             {rela["Sobrinho Paterno"]++};

        if($i ~ /,Primo/)
             {rela["Primo"]++};

        if($i ~ /,Avo Materno/)
             {rela["Avo Materno"]++};

        if($i ~ /,Avo Paterno/)
             {rela["Avo Paterno"]++};

        if($i ~ /,Tio Avo Materno/)
             {rela["Tio Avo Materno"]++};

        if($i ~ /,Tio Avo Paterno/)
             {rela["Tio Avo Paterno"]++};

        if($i ~ /,Neto Paterno/)
             {rela["Neto Paterno"]++};
```

Figura 4.3: Implementação da alinea c (parte 1)

```
46          if($i ~ /,Neto Paterno/)
47                {rela["Neto Paterno"]++};
48
49          if($i ~ /,Neto Materno/)
50                {rela["Neto Materno"]++};
51
52          if($i ~ /,Sobrinho Bisneto Paterno/)
53                {rela["Sobrinho Bisneto Paterno"]++};
54
55          if($i ~ /,Sobrinho Bisneto Materno/)
56                {rela["Sobrinho Bisneto Materno"]++};
57      }
58  }
59
60  END {
61       for (r in rela) {print "Da relação  " r "  há  " rela[r] "  relações"}
62       }
63
```

Figura 4.4: Implementação da alinae c (parte 2)

4.2.1 Execução do programa e Resultados obtidos

Para a execução do programa, criamos uma Makefile com os comandos necessarios. Esta makefile compila os 3 ficheiros ao mesmo tempo.

```
Tarefas:
        gawk -f expprocessosa.gawk < processos.txt
        gawk -f expprocessosc.gawk < processos.txt
        gawk -f expprocessosb.gawk < processos.txt
```

Figura 4.5: Makefile

Depois de executarmos a makefile obtemos os resultados pretendidos para cada alinea, como se pode ver seguidamente nas imagens.

Como o ficheiro "processos.txt"é enorme, os resultados obtidos tambem são enormes e por isso não conseguimos colocar aqui imagens dos resultados completos, apenas de partes.

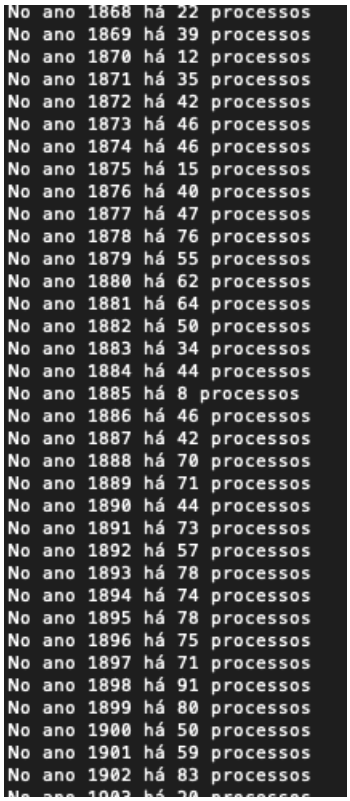

Figura 4.6: Resultado obtido da execução de "expprocessosa.gawk"

Nesta figura observa-se que obtivemos os anos existentes e respetivo número de processos nesse ano

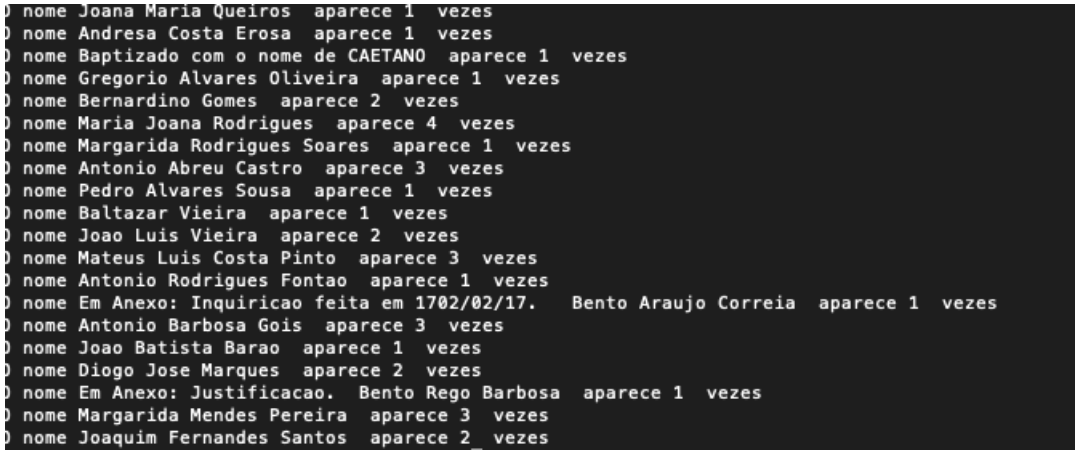

Figura 4.7: Resultado obtido da execução de "expprocessosb.gawk"

Nesta figura 4.6 encontram-se os resultados obtidos após a execução do codigo. No entano não é o resultado que esperavamos pois não conseguimos tirar frases que nao correspondem a nomes.

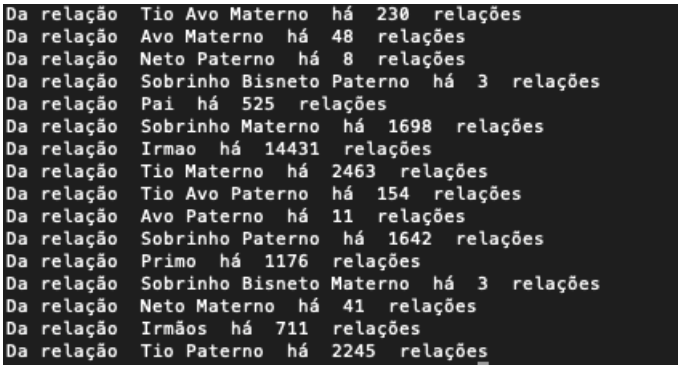

Figura 4.8: Resultado obtido da execução de "expprocessosc.gawk"

Nesta figura observa-se que obtivemos o numero de relações para cada tipo de relação existente.

Capítulo 5

Conclusão

O (G)Awk é uma linguagem e processador de padrões em texto. 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 mesmo nao tendo concluido uma das alineas com sucesso.

Apêndice A

Código do Programa

Lista-se de seguida o código GAWK do programa que foi desenvolvido para a alínea a) do problema.

```
EGIN {FS = "::"}

  {
  split($2,data,"-");
  count[data[1]]++;
  }

END {
for (ano in count) {print "No ano " ano " há " count[ano] " processos"};
}
```

Lista-se agora o código GAWK do programa que foi desenvolvido para a alinea b) do problema.

```
BEGIN{FS = "::"}
{
   for(i=3; i<NF-1; i++){
       {freq[$i]++};
    }
    {split($6,name,",");
    freq[name[1]]++;
    {split(name[1],name3,".")
      freq[name3[3]]++;}
    {split(name[2],name2,".");
      freq[name2[4]]++;}
    }
}

END {
     for (nome in freq) {print "O nome " nome "  aparece " freq[nome] "  vezes"}
     }
```

Lista-se a baixo o código GAWK do programa que foi desenvolvido para a alinea c) do problema.

```
BEGIN{FS = "."}
{
   for(i=1; i<=NF; i++){
```

```
    if($i ~ /,Irmaos/)
           {rela["Irmãos"]++};


    if($i ~ /,Irmao/)
           {rela["Irmao"]++};

    if($i ~ /,Tio Paterno/)
           {rela["Tio Patero"]++}


    if($i ~ /,Tio Materno/)
          {rela["Tio Materno"]++};


    if($i ~ /,Pai/)
          {rela["Pai"]++};

    if($i ~ /,Pais/)
          {rela["Pais"]++};

    if($i ~ /,Sobrinho Materno/)
          {rela["Sobrinho Materno"]++};

    if($i ~ /,Sobrinho Paterno/)
          {rela["Sobrinho Paterno"]++};

    if($i ~ /,Primo/)
          {rela["Primo"]++};

    if($i ~ /,Avo Materno/)
          {rela["Avo Materno"]++};

    if($i ~ /,Avo Paterno/)
          {rela["Avo Paterno"]++};

    if($i ~ /,Tio Avo Materno/)
          {rela["Tio Avo Materno"]++};

    if($i ~ /,Tio Avo Paterno/)
          {rela["Tio Avo Paterno"]++};

    if($i ~ /,Neto Paterno/)
          {rela["Neto Paterno"]++};

    if($i ~ /,Neto Materno/)
          {rela["Neto Materno"]++};

    if($i ~ /,Sobrinho Bisneto Paterno/)
          {rela["Sobrinho Bisneto Paterno"]++};

    if($i ~ /,Sobrinho Bisneto Materno/)
          {rela["Sobrinho Bisneto Materno"]++};
}
```

```
}

END {
     for (r in rela) {print "Da relação  " r "  há  " rela[r] "  relações"}
     }
```