Processamento de Linguagens e Compiladores

Processador de Róis de Confessados

Relatório de Desenvolvimento

André Sá (76361) João Bastos (47419) Pedro Sá (78164)

18 de Novembro de 2018

Resumo

O trabalho apresentado consiste na criação de um processador de texto que a partir de Róis de Confessados disponibilizados calcula a frequência de alguns padrões.

Conteúdo

Capítulo 1

Introdução

1.1 Enquadramento

Os róis de confessados eram listas nominativas elaboradas durante a quaresma a fim de constatar quais fiéis se confessaram e comungaram no último ano. Este texto determinava que nos róis constassem os nomes de todos os paroquianos, estado matrimonial, se eram crismados, se eram menores de idade, entre outras informações consideradas relevantes. Os róis de confessados deveriam listar os fregueses de cada domicílio, separando as informações com uma linha. O processador texto que iremos criar será implementado sobre um exemplo de rol de confessados.

1.2 Estrutura do Relatório

Neste relatório começamos com a Introdução (Capítulo 1) onde, de uma forma breve tentamos descrever o enquadramento do documento e a estrutura do mesmo.
No capítulo 2 analisamos o texto a processar e o problema que nos foi proposto de forma mais detalhada bem como uma introdução da nossa abordagem ao mesmo.
No capítulo 3 indicamos a estrutura de dados utilizada e explicamos o código por nós criado. O capítulo 4 apresentamos alguns testes realizados e o respectivo output. O capítulo 5 é composto pela conclusão, onde fazemos uma análise ao projecto executado. Por último o documento tem o apêndice onde foi introduzido o código fonte do ficheiro Gawk.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Este projecto tem como objectivo o aprofundamento de conhecimento do Sistema de Produção[1] AWK (uma lingagem de programação para a deteção de padrões e processamento de texto, mais o respetivo interpretador). Para tal, foi-nos proposto construir um ficheiro Gawk[2] para processar um rol de confessados, com o intuito de calcular o número de certas ocorrências.
Na próxima secção indicaremos com mais detalhe as ocorrências pretendidas e de que forma pretendemos resolver tais problemas.

2.2 Especificação dos Requisitos

O Rol de Confessados que nos foi facultado tem um registo por linha. Cada campo do registo está separado por "::", sendo que 6 têm o seguinte conteúdo:

1º campo ($1) Número do registo;

2º campo ($2) Data do registo;

3º campo ($3) Nome do paroquiano a registar;

4º campo ($4) Pai do paroquiano;

5º campo ($5) Mãe do paroquiano;

6º campo ($6) Informação complementar.

e o último campo apresenta-se sempre vazio.

Todos os registos têm os primeiros 3 campos preenchidos, podendo os restantes estar omissos. Apresentemos o exemplo de um registo completo para facilitar na compreensão do exposto acima e nas seguintes referências a registos do Rol de Confessados.

```
598::1722-07-29::Pedro Jose Sousa::Pedro Sousa Falcao::Mariana Sousa Pereira
::Manuel Felix Sousa,Irmao. Proc.8224.::
```

Foi-nos proposto sobre o Rol de Confessados fazermos o seguinte processamento:

[1] Sistema baseado num conjunto de regras 'Condição-Acção

[2] Gawk - GNU AWK (A palavra AWK é uma abreviatura das iniciais dos sobrenomes dos criadores da linguagem (Aho, Weinberger e Kernighan).

Alínea a) - Calcular a frequência de processos por ano

Dentro do 2ª campo, que representa a data do registo, teremos de verificar em cada registo o ano indicado e calcular o número de vezes que cada ano aparece.

Alínea b) - Calcular a frequência de nomes

Dentro dos campos 3, 4 e 5 teremos de verificar todos os nomes apresentados e contabilizá-los para após todas as linhas serem processadas apresentarmos o número de ocorrências de cada nome em todas as linhas do rol.

Alínea c) - Calcular a frequência dos vários tipo de relação

Em alguns registos existe informação sobre os tipos de relação existentes com o paroquiano em questão. Essa informação está presente somente no 6º campo pelo que teremos de verificar as ocorrências dessas relações e contabilizar todas para no final apresentarmos o número de ocorrências de cada relação.

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

Um array associativo é uma estrutura de dados - análoga aos dicionários - composta por um conjunto não ordenado de itens formados por um par chave e valor, em que cada chave possui um valor associado.
O Gawk permite-nos utilizar strings como índices. Assim, optamos pela utilização de arrays associativos para contar a ocorrência de determinadas strings sendo as strings as chaves e o valor correspondente ao número de ocorrências de cada uma.

3.2 Codificação

BEGIN

Inicialmente especificamos acções a serem executadas antes do processamento da primeira linha utilizando o padrão BEGIN.

```
BEGIN   {
            FS="::";
            IGNORECASE=1
        }
```

Assim definimos o Field Separator (FS) como "::" e a detecção de padrões como Case Insensitive.

Linhas em Branco

Salvaguardamos o facto do ficheiro ter linhas em branco ignorando as mesmas.

```
/^\s*$/ { next; }
```

Alínea a) - Frequência de processos por ano

Para calcularmos o número de processos por ano separamos o que está entre '-' no 1º campo e, num array associativo, utilizamos o elemento correspondente ao ano[1] como chave. Se a chave já existe apenas incrementa, caso contrário cria a nova chave e incrementa o valor.

```
{
     split($2, data, /-/);
     alA[data[1]]++;
}
```

[1]Formato do 1º campo: 1778-01-14. Ter em conta que em Awk os índices dos arrays sendo numéricos começam em 1, que é o índice onde será guardado o ano no array data.

Alínea b) - Frequência de nomes

Os nomes que pretendemos vão estar todos entre os campos 3, 4 e 5, que correspondem respectivamente ao paroquiano, o seu pai e a sua mãe. Nos campos indicados estão os nomes completos mas, uma vez que pretendemos contar as ocorrências de cada nome, separamos os vários nomes e apelidos e guardamos num array associativo. Cada nome/apelido será a chave desse array e a cada ocorrência irá incrementar o respectivo valor. Assim, no final do processamento de todas as linhas do texto teremos guardado nesse array todos os nomes/apelidos (chave) e o número de ocorrências de cada um deles (valor).

```
{
    for (i = 3; i < 6; i++) {
        # Tirar Expostos/Solteiras/etc
        gsub(/,.*$/, "", $i);

        split($i, nomes, /\s+/);

        for (nome in nomes)
            alB[nomes[nome]]++;
    }
}
```

Alínea c) - Frequência dos tipos de relação

Analisando o rol de confessados que nos foi facultado verificamos que dentro do 6º campo temos informação complementar onde ,entre outras informações, surgem os tipos de relação ou grau de parentesco. Portanto procuramos correspondências com os padrões dentro do 6º campo de cada linha. Verificamos também que cada relação identificada é precedida de uma virgula e termina com o ponto final. Assim, optamos pela procura desse padrão associando a correspondência à chave do array associativo incrementando o seu valor a cada correspondência obtida.

```
$6 ~ /,\<avo\>([ ]+[mp]atern[oa]s?)?\./        { alC["avo"]++; }
$6 ~ /,\<irmaos?\>([ ]+[mp]atern[oa]s?)?\./    { alC["irmao"]++; }
$6 ~ /,\<irmas?\>([ ]+[mp]atern[oa]s?)?\./     { alC["irma"]++; }
$6 ~ /,\<mae\>\./                              { alC["mae"]++; }
$6 ~ /,\<madrastas?\>([ ]+[mp]atern[oa]s?)?\./ { alC["madrasta"]++; }
$6 ~ /,\<padrastos?\>([ ]+[mp]atern[oa]s?)?\./ { alC["padrasto"]++; }
$6 ~ /,\<madrinhas?\>([ ]+[mp]atern[oa]s?)?\./ { alC["madrinha"]++; }
$6 ~ /,\<padrinhos?\>([ ]+[mp]atern[oa]s?)?\./ { alC["padrinho"]++; }
$6 ~ /,\<pai\>\./                              { alC["pai"]++; }
$6 ~ /,\<primas?\>([ ]+[mp]atern[oa]s?)?\./    { alC["prima"]++; }
$6 ~ /,\<primos?\>([ ]+[mp]atern[oa]s?)?\./    { alC["primo"]++; }
$6 ~ /,\<sobrinhas?\>([ ]+[mp]atern[oa]s?)?\./ { alC["sobrinha"]++; }
$6 ~ /,\<sobrinhos?\>([ ]+[mp]atern[oa]s?)?\./ { alC["sobrinho"]++; }
$6 ~ /,\<tias?\>([ ]+[mp]atern[oa]s?)?\./      { alC["tia"]++; }
$6 ~ /,\<tios?\>([ ]+[mp]atern[oa]s?)?\./      { alC["tio"]++; }
```

Funções

Definimos 3 funções que irão escrever o conteúdo de cada um dos arrays associativos - cada um no seu ficheiro - de forma formatada e com uma descrição na primeira linha de cada um.

```
function alineaA (outf)
{
    printf "Frequência de processos por ano:\n" > outf;
    for (ano in alA)
```

```
        printf("%s\t%s\n", ano, alA[ano]) >> outf;
}

function alineaB (outf)
{
    printf "Frequência de nomes:\n" > outf;
    for (nome in alB)
        printf("%s\t%s\n", nome, alB[nome]) >> outf;
}

function alineaC (outf)
{
    printf "Frequência dos vários tipos de relação:\n" > outf;
    for (parentesco in alC)
        printf("%s\t%s\n", parentesco, alC[parentesco]) >> outf;
}
```

END

Para finalizar utilizamos o END para executarmos as funções acima definidas após todas as linhas terem sido executadas.

```
END     {
          alineaA("anos.txt");
          alineaB("nomes.txt");
          alineaC("parentescos.txt");
        }
```

Criando assim os 3 ficheiros com extensão txt que irão armazenar os resultados pedidos para cada uma das alíneas.

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

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

Bibliografia