

Trabalho Final de ICC (2017/02)

Pablo Cecilio, Marco Antônio, Lucas Souza

1 Introdução

O trabalho consiste no processamento e manuseio de uma base de dados usando a programação em bash.

Para esse processamento foi utilizado um script contendo os comandos necessários para executar ações específicas requisitadas pelos itens abaixo descritos.

2 Pré-processamento

Para realizar as extrações solicitadas, os dois arquivos “title.basics.tsv” e “title.ratings.tsv” foram pré-processados utilizando o comando “join”, que concatena linhas de dois arquivos através de uma coluna em comum.

```
join -t $'\t' -o 2.1,2.2,2.3,2.4,2.5,2.6,2.7,2.8,2.9,1.2,1.3  
↪ title.ratings.tsv title.basics.tsv >  
↪ ../$OUTPUTDIR/titles.tsv
```

Após a execução do comando, o arquivo “titles.tsv” foi gerado na pasta de saída passada por argumento na chamada do script.

Para finalizar o pré-processamento, o comando “sed” foi utilizado, removendo a primeira linha do arquivo “titles.tsv” e gerando um novo arquivo, “titles.all.tsv”.

```
sed '1d' ../$OUTPUTDIR/titles.tsv >  
↪ ../$OUTPUTDIR/titles.all.tsv
```

De modo a conferir se as saídas foram geradas corretamente, o comando “wc -l” foi executado ao final do processo para comparação de seus resultados com o numero de linhas entre os arquivos “titles.ratings.tsv” e o arquivos gerados, “titles.tsv” e “titles.all.tsv”.

```
echo -e "#Linhas nos arquivos originais:"  
wc -l ../$IMBDDIR/title.basics.tsv  
wc -l ../$IMBDDIR/title.ratings.tsv  
echo -e "\n#Linhas nos arquivos gerados:"  
wc -l titles.tsv  
wc -l titles.all.tsv
```

3 Extrações

A extração dos itens 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17 e 18 foram realizadas.

Item 1

O comando “cut” foi utilizado para selecionar a segunda coluna em “title.all.tsv”. Após essa ação, foi realizado um pipe para “sort” afim de ordenar essa seleção e em seguida outro pipe para “uniq”, com isso listando as entradas únicas como saída para o arquivo “out1”.

```
cut -f 2 titles.all.tsv | sort | uniq | tee out1
```

ENTRADA	CUT -F 2	SORT	UNIQ
F A X	A	A	A
G B Y	B	A	B
H A Z	A	B	C
K C U	C	C	

Item 2

Utilizou-se o comando “awk” com uma variável como contador, essa por sua vez foi condicionada a aumentar seu valor caso as colunas 3 e 4 fossem iguais. A saída foi impressa em arquivo com o resultado da variável.

```
awk -F"\t" '{if ($3 == $4) s += 1} END{print s}' titles.all.tsv  
↪ | tee out2
```

Item 3

Por meio do comando “awk”, comparou-se a coluna 6 com os valores 1970 a 2000, e para cada resultado positivo uma variável somava a nota dos títulos a um total, enquanto outra variável recebia a quantidade de filmes da condição. Ao final da execução, o total das notas foi dividido pela quantidade de títulos, obtendo-se a média.

```
awk -F"\t" '{if (1969 < $6 && $6 < 2001){s += $10; t+= 1}}  
↪ END{printf("%.4f\n", s/t)}' titles.all.tsv | tee out3
```

Foi observado uma discrepância entre os valores gerados em sistemas diferentes, que foi atribuída a erros com operações com ponto flutuante devido a diferentes versões ou funções do bash. Como solução, um algoritmo foi acrescentado de forma comentada no script para demonstrar a correção por meio do uso de variáveis e pelo comando “bc”.

```
# soma=$(awk -F"\t" '{if ($6>=1970 && $6<=2000){print $10}}'
↪ titles.all.tsv | paste -sd+ - | bc)
# total=$(awk -F"\t" '{if ($6>=1970 && $6<=2000){count++}}'
↪ END{print count}' titles.all.tsv)
# bc <<< "scale=10;$soma / $total"
```

Item 4

Segue a mesma lógica e princípio do item 3, apresentando a mesma solução e o mesmo erro relatado no item anterior.

```
awk -F"\t" '{if (1999 < $6 && $6 < 2017){s += $10; t+= 1}}'
↪ END{printf("%.4f\n", s/t)}' titles.all.tsv | tee out4
```

Item 5

A coluna 9 do arquivo foi separada, e com uma inversão de busca (“grep -v”) eliminou-se as linhas que possuem ‘,’ e ‘\N’. Após o pipe, a saída foi organizada (“sort”) por ordem alfabética e as entradas repetidas foram removidas com “uniq”. Por fim, o resultado foi gerado contando-se as linhas dessa saída com o comando “wc -l”, conforme o proposto.

Diversos erros de sintaxe foram apresentados no desenvolvimento ao usar o “grep -v” no \N sem utilizar \\.

Como detalhe a parte do proposto, foi acrescentado um “tr -d” para cortar *whitespaces* gerados pela saída do comando “wc” no macOS. Esses espaços em branco eram antes gravados na saída.

```
cut -f 9 titles.all.tsv | grep -v "," | grep -v "\\N" | sort |
↪ uniq | wc -l | tr -d ' ' | tee out5
```

Item 6

Utilizou-se “cut -f 9” para separar a coluna 9 da base de dados no arquivo, após essa ação, foi feita a busca pela *string* “Action”. O resultado foi gerado ao contar as linhas desse saída.

```
cut -f 9 titles.all.tsv | grep "Action" | wc -l | tr -d ' ' |  
↪ tee out6
```

Item 7

O retorno da saída foi gerado pelo “awk”, que tinha como condição encontrar a *string* “Adventure” na coluna 9 e o ano maior que 2005, ignorava as entradas com “\N”. Novamente foi usado o “wc l” como contador.

```
awk -F"\t" '$9 ~ /Adventure/ && $6 != "\\N" && $6 >= 2005  
↪ {print $6,"\t",$9,"\t",$3}' titles.all.tsv | sort | wc -l |  
↪ tr -d ' ' | tee out7
```

Item 8

Usa a mesma lógica apresentada no item 7, porém, com condição e operadores lógicos diferentes para a solução.

```
awk -F"\t" '{if (($9 ~ /Fantasy/ || $9 ~ /Sci-Fi/) && ($6 !=  
↪ "\\N" && $6 >= 2010)) print $6,"\t",$9,"\t",$3}'  
↪ titles.all.tsv | sort | wc -l | tr -d ' ' | tee out8
```

Item 9

Uma variável foi utilizada para guardar o resultado gerado pela seleção e contagem de itens com o ano igual a 1970 (Contagem feita no mesmo

procedimento de itens anteriores). Após o valor ter sido gerado, a razão foi obtida pela divisão desse valor pelo número total de títulos na base já guardados em uma variável anterior no script . Para essa operação “bc” foi utilizado a fim de gerar o número de ponto flutuante.

```
item9=$(awk -F"\t" '{if ($6 == 1970 && $6 != "\\N") print $6}'  
↪ titles.all.tsv | wc -l | tr -d ' ' )  
bc <<< "scale=5;$item9 / $titulos" | tee out9
```

Item 10

Foi usada a mesma solução do item 9, porém um laço foi utilizado para gerar por cada ano a razão desse ano pelo total de títulos entre 1971 e 2016.

Uma variável foi gerada para guardar cada ano a ser utilizado no laço, e esse usava o ano dessa variável a fim de selecionar o total de títulos e obter a cada passagem a razão respectiva, já guardada em outra variável (“\$range”).

```
anos=$(cut -f 6 titles.all.tsv | sort | uniq | sed '$d')  
range=$(cut -f 6 titles.all.tsv | awk '$NF >= 1971 && $NF <=  
↪ 2016' | wc -l | tr -d ' ' )  
echo Titulos produzidos no intervalo 1971-2016: $range  
for i in $anos;  
do  
    item10=$(cut -f 6 titles.all.tsv | grep -c "$i")  
    media10=$(bc <<< "scale=5;$item10 / $range")  
    echo -e "$i\t"$media10 | tee -a out10  
    item10=  
done
```

Há dúvidas a respeito dos resultados gerados, mesmo esses sendo obtidos pelo comando “bc”. Os números gerados por meio do laço não coincidem com o gabarito parcial do problema proposto.

Item 11

Após separar a coluna de “genres”, a busca por “,” foi invertida gerando o total dado pelas contagem de linhas desse resultado. Como padrão, “\N” também foi ignorado quando utilizado o mesmo método de busca invertida.

```
cut -f 9 titles.all.tsv | grep -v "," | grep -v '\\N' | wc -l |  
↪ tr -d ' ' | tee out11
```

Item 12

Por meio de “cut”, separo-se a coluna 9. Após essa ação, o comando “tr” foi usado para trocar as “,” por “\n”, separando assim as linhas que possuíam mais de um gênero. Logo após a ordenação dos itens através do “sort”, o “uniq -c” foi usado para a contagem de itens iguais.

Para a finalizar a extração, uma busca invertida foi usada em “\N”, com a saída desse comando sendo ordenada com o “sort” reverso através do pipe.

```
cut -f 9 titles.all.tsv | tr -s "," "\n" | sort | uniq -c |  
↪ grep -v "\\N" | sort -g -r | tr -d [:digit:] | tr -d ' ' |  
↪ head -n 5 | tee out12
```

Nota: Para fins de formatação foi acrescentado o código abaixo, que remove os números e espaços da saída.

```
tr -d [:digit:] | tr -d ' '
```

Item 16

Utiliza a mesma lógica apresentada nos itens 9 e 10 para gerar o resultado.

```
item16a=$(cut -f 8 titles.all.tsv | grep -v "\\N" | awk '$NF >=
↪ 80 && $NF <= 120' | wc -l | tr -d ' ')
item16b=$(cut -f 8 titles.all.tsv | grep -c -v "\\N")
bc <<< "scale=5;$item16a / $item16b" | tee out16
```

Item 17

Foi usado o condicionamento no “awk” para gerar uma saída a fim de ordena-la com “sort” pelo parâmetro da segunda coluna. Por fim, o comando “head” foi utilizado para listar os dez primeiros itens da saída

```
awk -F"\t" '{if(($9 ~ /Action/) && ($6 >= 2005 && $6 != "\N")
↪ && ($2 ~ /movie/) && ($11 > 100)) print $3,"\t",$10}'
↪ titles.all.tsv | sort -r -g -t'\t' -k2 | head -n 10 | tr
↪ '\t' ' ' | tee out17
```

Nota: Usou-se “tr” para a retirada de espaços para o pipe no arquivo de saída.

Item 18

Usa a mesma lógica apresentada no item 17, porém, com condição e operadores lógicos diferentes para a solução.

```
awk -F"\t" '{ if (($2 ~ /movie/) && ($9 ~ /Comedy/) && ($11 >
↪ 100) && ($8 > 200)) print $3,"\t",$10}' titles.all.tsv |
↪ sort -r -g -t'\t' -k2 | head -n 5 | tr '\t' ' ' | tee
↪ out18
```


4 Organização do trabalho

O processo de colaboração do trabalho foi realizado e documentado utilizando o *GitHub* como plataforma. https://github.com/Durfan/ICC_TP/

A comunicação imediata, assim como o planejamento, foi feito de forma mais informal por meio do *WhatsApp*. Sendo realizado através desse aplicativo a divisão e designação de tarefas por demanda ou por forma voluntária.

MEMBRO	ATIVIDADES
Pablo Cecilio	pré-processamento; script.sh; itens 1, 7, 8, 9, 10, 16; documentação
Marco Antônio	itens 2, 3, 4, 5, 6 e 11
Lucas Souza	itens 12, 17 e 18
Arthur Rocha	—

5 Conclusão

A falta de uma documentação bash didática e menos técnica dificultou e atrasou o desenvolvimento, sendo que a pesquisa realizada para resolver cada problema enunciado foi feita na maior parte por soluções diversas encontradas em fóruns e sites como o stackoverflow.com. Somando-se a isso, a maior dificuldade encontrada foi em relação a sintaxe dos comandos, esses que por sua vez variam na forma de suas expressões lógicas e possuem particularidades tais como o uso de aspas simples ou duplas. Ex: O uso de *rege~~re~~xp* para o comando egrep (grep -E), que podia realizar com exatidão metade das buscas enunciadas, porém, sua implementação foi descartada por problemas na sintaxe não correspondendo a solução.

Durante o desenvolvimento também foram encontradas discrepâncias em sistemas diferentes com operações envolvendo ponto flutuante. Sendo que, particularmente, algoritmos envolvendo divisões em sistemas 64bits retornam valores diferentes em relação a outros sistemas, diferindo do problema enunciado. Além disso, o tempo de execução do script varia não apenas devido as especificações da máquina, mas também devido a diferentes versões do bash e ao módulo math incorporado.

6 Extra

