universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Estatísticas de utilizadores em bash

Licenciatura em Engenharia Informática
Sistemas Operativos
Docente: Nuno Lau

Isadora Loredo, 91322
Gonçalo Matos, 92972

Ano letivo 2019/2020

Índice

Introdução

Neste projeto foi-nos proposto que desenvolvêssemos dois *scripts* em *bash* para a recolha de algumas estatísticas sobre o modo como os utilizadores utilizam o sistema operativo, nomeadamente o número de sessões, os tempos que estas duraram e algumas relações entre estes e outras variáveis como o grupo a que pertencem, que serão explicadas em detalhe neste relatório.

Apresentados nesta ordem, o primeiro é responsável pela consulta e processamento dos dados, transformando-os em informação que é impressa no terminal e pode ser escrita num ficheiro, que posteriormente pode ser processado pelo segundo, que tem como objetivo comparar dois ficheiros resultantes da execução do primeiro e mostrar ao utilizador a relação entre a informação presente em ambos.

Todo o código foi desenvolvido de raiz, tendo por base os conhecimentos adquiridos nas aulas práticas e a pesquisa realizada durante o decorrer do projeto, e como objetivo a correspondência dos resultados com o *output* esperado, descrito no guião do projeto, com vários exemplos de execução.

Soluções implementadas

userstats.sh

A execução deste *script* começa por inicializar algumas variáveis, cujo propósito será descrito mais à frente aquando da sua utilização, seguido do processamento das opções de execução introduzidas pelo utilizador.

Para as opções que aceitam argumentos (-g, -u, -s e -f), é lido o argumento (elemento seguinte do *array* que guarda os argumentos - *${args[a + 1]}*) e para as restantes opções que são de filtragem (-r, -n, -t, -a e -i) é verificado se não estão a ser adicionadas opções incompatíveis (apenas pode existir uma opção -n, -t, -a ou -i, que podem ser conjugadas (opcionalmente) com a opção -r e caso isto não se verifique, são adicionadas à variável de filtragem.

De seguida são invocadas duas funções, uma que cria uma listagem dos utilizadores que vão ser processados (tendo em conta os utilizadores que constam nos dados disponíveis para consulta e os filtros aplicados) - *countUsers()* - e outra para processar os detalhes associados às sessões de cada utilizador listado anteriormente - *detailSessions()*.

Verifica-se ainda qual Sistema Operativo está a correr o *script* entre *Linux* e *macOS*, uma vez que alguns comandos em *bash* possuem diferenças para esses dois sistemas. Esta verificação é aplicada nas funções desenvolvidas, para garantir que ambos os membros do grupo pudessem correr e testar as funcionalidades implementadas.

countUsers()

Esta função começa por verificar se é necessário haver filtragem dos utilizadores por grupo, através do acesso ao ficheiro */etc/group* do sistema, que é filtrado em busca do nome do grupo passado como argumento, obtendo a lista de utilizadores pertencentes ao mesmo. Caso não haja filtragem por grupo, os utilizadores a serem processados serão todos para os quais existam registos, sendo estes dados obtidos através da consulta da primeira coluna do *output* do comando *last* (que corresponde ao número de utilizadores), removendo os repetidos e ainda alguns utilizadores que estão associados a processos do sistema e que não achamos pertinente o seu *output* (nomeadamente o *shutdown*, *reboot*, *root* e *_mbsetupuser*). Em ambos os cenários descritos, a lista dos utilizadores é armazenada na variável *users*.

Por fim, caso seja necessária a filtragem por utilizador, o *array* criado no processo anterior será percorrido e os elementos que não correspondem à expressão *regex* passada como argumento serão eliminados do mesmo.

Finalmente está concluída a função e a lista de utilizadores cujos dados das sessões serão analisados está compilada.

detailSessions()

Esta função seleciona as informações estatísticas a serem mostradas (número de sessões e tempos total, máximo e mínimo de ligação das sessões), por utilizador da variável *users*, sendo estes os utilizadores definidos pela função *countUsers()* através da filtragem realizada de acordo com os argumentos passados.

Para cada utilizador em *users* primeiramente é realizado o filtro do período temporal, estes feitos através do comando *last* (-s <data> -t <data>), se foi passado como argumentos as datas de início da sessão a partir da qual as sessões devem ser consideradas (opção -s) e da data de início de sessão a partir da qual as sessões não devem ser consideradas (opção -e). Não sendo feita a inserção de um ou nenhum dos argumentos para o filtro temporal as datas consideradas são as que que houverem de registo do ficheiro */var/log/wtmp* por defeito ou de algum outro ficheiro quando esse for passado por como argumento (opção -f). Todos os registos são lidos, de acordo com os critérios citados e processados, e cada tempo de sessão é armazenado em um *array*, variável *time*, e de seguida é percorrido para determinar os tempos total, máximo e mínimo de sessões, em minutos. O número de sessões é determinado pela quantidade de entradas que possui a variável *time*.

Essa função constrói a variável *output*, que é o conjunto de informações compiladas a ser apresentado na consola.

dateConversion()

Esta função converte o formato das datas passadas como argumentos das opções -s e -e que realizam o filtro do período temporal, para um formato aceito pelo comando *last*.

comparestats.sh

A execução deste *script* começa por validar a quantidade mínima de argumentos passados, e a existência dos ficheiros a serem comparados. De seguida são inicializadas algumas variáveis, cujo propósito será descrito mais à frente aquando da sua utilização.

O ficheiro passado como primeiro argumento, a representar os valores mais recentes, é lido linha-a-linha. Cada linha deste ficheiro corresponde a um utilizador, e é feita a verificação se este utilizador armazenado na variável *userF1* também aparece no outro ficheiro, passado no segundo argumento. Havendo a correspondência, em que o mesmo utilizador se encontra em ambos os ficheiros, as diferenças para os dados de número de sessões e tempos total, máximo e mínimo são calculadas entre o valor do primeiro e segunda ficheiro e estes dados são então armazenados na variável *output* e este usuário catalogado, na variável *usedUsers*. Em caso de não correspondência, onde o usuário encontra-se apenas no primeiro ficheiro, o cálculo das diferenças não são feitos e os dados desse utilizador são integralmente copiados para a variável *output*.

Assim que concluída a leitura do primeiro ficheiro dá-se início à leitura do segundo ficheiro, também linha-a-linha. Para cada utilizador deste ficheiro, armazenado na variável *userF2*, é realizada a verificação se este não estava no primeiro ficheiro, se esse utilizador não se encontrar na variável *usedUsers*. Sendo a verificação válida, os dados deste utilizador são adicionados à variável *output*.

Por fim, os argumentos são processados para verificar se foi solicitada a ordenação na visualização da informação mostrada na consola, opções -r (ordem decrescente), -n (por número de sessões), -t (por tempo total), -a (por tempo máximo) e -i (por tempo mínimo).

Por fim é apresentada a informação compilada na variável *output*, de forma ordenada se assim solicitado.

Conclusão

A realização deste projeto permitiu-nos aprofundar os conhecimentos sobre a *bash*, tendo as bases aprendidas nas aulas práticas sendo o ponto de partida para a compreensão dos comandos que necessitámos de utilizar.

Descobrimos que a compreensão dos comandos da *bash*, apoiada pela extensa documentação e ajuda disponibilizada *online*, é mais fácil do que esperávamos e sentimos agora estar mais aptos para abraçar desafios com que nunca nos deparámos na utilização desta *shell*.

Apêndice A: Testes de validação

Os ficheiros utilizados e criados durante a realização dos testes descritos abaixo serão submetidos em conjunto com o projeto, no diretório *remoteData*.

userstats.sh

Para testar este *script* acedemos remotamente ao computador sop0109@l040101-ws05, por forma a obter resultados semelhantes aos sugeridos pelo professor no guião do projeto.

```
./userstats.sh
nlau    1 0   0 0
sop0102    2 0 0    0
sop0104    4 0 0    0
sop0109    5 0 0    0
sop0205    22 1942   397 1
sop0207    25 151   96 1
sop0210    27 35 33    2
sop0304    34 226   138 2
sop0405    38 150   131 0
sop0408    64 3383   401 0
sop0410    93 2936   418 0
```

```
./userstats.sh -f /var/log/wtmp.1
```

Não conseguimos executar este comando no computador remoto, pois o ficheiro não existia no mesmo. Em alternativa corremos no nosso computador pessoal. O *output* obtido foi o seguinte:

```
goncalo    131 10177    283 0
```

```
./userstats.sh -u “sop.*”
sop0102    1 0 0    0
sop0104    3 0 0    0
sop0109    8 40 40    0
sop0205    25 1942   397 1
sop0207    28 151   96 1
sop0210    30 35 33    2
sop0304    37 226   138 2
sop0405    41 150   131 0
sop0408    67 3383   401 0
sop0410    96 2936   418 0
```

```
./userstats.sh -g sop
```

O grupo ‘sop’ não tinha utilizadores atribuídos aquando da realização dos testes. Por isso, fizémos a filtragem pelo grupo ‘wheel’, o único grupo com utilizadores atribuídos (‘ruib,nlau’ - apenas o último tem registo de sessões) no ficheiro *wtmp*. O output obtido foi o seguinte:

```
nlau    1 0   0 0
```

```
./userstats.sh -s “Nov 19 10:00” -e “Nov 23 18:00”
sop0205    5 566 397    1
sop0304    10 72 23    2
sop0408    12 260   260 0
sop0410    15 55 49    2
```

```
./userstats.sh -t -u “sop.*”
sop0102    1 0 0    0
sop0104    3 0 0    0
sop0210    30 35 33    2
sop0109    8 40 40    0
```

```
sop0405    41 150   131 0
sop0207    28 151   96 1
sop0304    37 226   138 2
sop0205    25 1942   397 1
sop0410    96 2936   418 0
sop0408    67 3383   401 0
```

```
./userstats.sh -n -u “sop.*”
sop0102    1 0 0    0
sop0104    3 0 0    0
sop0109    8 40 40    0
sop0205    25 1942   397 1
sop0207    28 151   96 1
sop0210    30 35 33    2
sop0304    37 226   138 2
sop0405    41 150   131 0
sop0408    67 3383   401 0
sop0410    96 2936   418 0
```

```
./userstats.sh -t -r -u “sop.*”
sop0408    67 3383   401 0
sop0410    96 2936   418 0
sop0205    25 1942   397 1
sop0304    37 226   138 2
sop0207    28 151   96 1
sop0405    41 150   131 0
sop0109    8 40 40    0
sop0210    30 35 33    2
sop0104    3 0 0    0
sop0102    1 0 0    0
```

```
./userstats.sh -a -r -u “sop.*”
sop0410    96 2936   418 0
sop0408    67 3383   401 0
sop0205    25 1942   397 1
sop0304    37 226   138 2
sop0405    41 150   131 0
sop0207    28 151   96 1
sop0109    8 40 40    0
sop0210    30 35 33    2
sop0104    3 0 0    0
sop0102    1 0 0    0
```

```
./userstats.sh -i -r -u “sop.*”
sop0304    37 226   138 2
sop0210    30 35 33    2
sop0207    28 151   96 1
sop0205    25 1942   397 1
sop0410    96 2936   418 0
sop0408    67 3383   401 0
sop0405    41 150   131 0
sop0109    8 40 40    0
sop0104    3 0 0    0
sop0102    1 0 0    0
```

```
./userstats.sh -n “.*” > userstats1
```

Para obter um ficheiro comparável e tendo em conta que o computador não tem o ficheiro */var/log/wtmp.1*, corremos o comando anterior no computador sop0109@l040101-ws06, uma vez que os utilizadores que existem em um, também existem no outro (há 7 utilizadores com informação acerca das suas sessões nos dois computadores), tendo gerado o ficheiro *userstats2*.

comparestats.sh

O teste deste *script* foi feito no computador sop0109@l040101-ws05, com os ficheiros gerados da forma descrita anteriormente.

```
$./comparestats.sh userstats1 userstats2
nlau    -1 -46    -32 -14
sd0104    -7 -719    -230 -82
sd0105    -16 -100    -44 0
sd0108    -17 -1   -1 -1
sd0109    -19 -87    -50 -37
sd0301    -79 0   0 0
sd0302    -190 -708    -133 0
sd0303    -218 -3    -2 0
sd0304    -219 0   0 0
sd0305    -238 0   0 0
sd0401    -259 0   0 0
sd0402    -349 -131    -131 0
sd0403    -350 -7    -7 -7
sd0405    -961 -45    -10 0
sd0406    -1127 -283    -131 0
sd0407    -1129 -187    -186 -1
sop0102    -1138 -242    -133 0
sop0104    -1137 0   0 0
sop0106    -1153 -146    -71 0
sop0109    -1155 -40    -9 0
sop0205    26 1942   397 1
sop0207    29 151   96 1
sop0210    -1149 -455    -183 2
sop0212    -1183 -208    -109 -27
sop0304    38 226   138 2
sop0310    -1199 -2486    -548 0
sop0311    -1208 -2910    -1275 0
sop0402    -1212 -926    -349 -123
sop0405    -1186 -2676    -360 0
sop0406    -1233 -22    -8 0
sop0407    -1343 -4944    -334 0
sop0408    -1309 -138    163 0
sop0409    -1378 0   0 0
sop0410    97 2936   418 0

$./comparestats.sh -r userstats1 userstats2
sop0410    97 2936   418 0
sop0409    -1378 0   0 0
sop0408    -1309 -138    163 0
sop0407    -1343 -4944    -334 0
sop0406    -1233 -22    -8 0
sop0405    -1186 -2676    -360 0
sop0402    -1212 -926    -349 -123
sop0311    -1208 -2910    -1275 0
sop0310    -1199 -2486    -548 0
sop0304    38 226   138 2
sop0212    -1183 -208    -109 -27
sop0210    -1149 -455    -183 2
sop0207    29 151   96 1
sop0205    26 1942   397 1
sop0109    -1155 -40    -9 0
sop0106    -1153 -146    -71 0
sop0104    -1137 0   0 0
sop0102    -1138 -242    -133 0
sd0407    -1129 -187    -186 -1
sd0406    -1127 -283    -131 0
sd0405    -961 -45    -10 0
sd0403    -350 -7    -7 -7
sd0402    -349 -131    -131 0
sd0401    -259 0   0 0
sd0305    -238 0   0 0
sd0304    -219 0   0 0
sd0303    -218 -3    -2 0
sd0302    -190 -708    -133 0
```

```
sd0301    -79 0   0 0
sd0109    -19 -87    -50 -37
sd0108    -17 -1   -1 -1
sd0105    -16 -100    -44 0
sd0104    -7 -719    -230 -82
nlau    -1 -46    -32 -14
```

```
$./comparestats.sh -t userstats1 userstats2
sop0407    -1343 -4944    -334 0
sop0311    -1208 -2910    -1275 0
sop0405    -1186 -2676    -360 0
sop0310    -1199 -2486    -548 0
sop0402    -1212 -926    -349 -123
sd0104    -7 -719    -230 -82
sd0302    -190 -708    -133 0
sop0210    -1149 -455    -183 2
sd0406    -1127 -283    -131 0
sop0102    -1138 -242    -133 0
sop0212    -1183 -208    -109 -27
sd0407    -1129 -187    -186 -1
sop0106    -1153 -146    -71 0
sop0408    -1309 -138    163 0
sd0402    -349 -131    -131 0
sd0105    -16 -100    -44 0
sd0109    -19 -87    -50 -37
nlau    -1 -46    -32 -14
sd0405    -961 -45    -10 0
sop0109    -1155 -40    -9 0
sop0406    -1233 -22    -8 0
sd0403    -350 -7    -7 -7
sd0303    -218 -3    -2 0
sd0108    -17 -1   -1 -1
sd0301    -79 0   0 0
sd0304    -219 0   0 0
sd0305    -238 0   0 0
sd0401    -259 0   0 0
sop0104    -1137 0   0 0
sop0409    -1378 0   0 0
sop0207    29 151   96 1
sop0304    38 226   138 2
sop0205    26 1942   397 1
sop0410    97 2936   418 0
```

Apêndice B: Comandos bash

Resumo dos comandos bash utilizados nos scripts desse projeto, adicionais aos comandos estudados nas aulas práticas.

COMANDO	PARÂMETROS	DESCRIÇÃO
grep	<expressão regular>	faz buscas no conteúdo de um ficheiro a procurar linhas que encontram a expressão regular mencionada
cut	-d <delimitador>; -f <coluna>	divide uma dada *string* por um delimitador; retorna a coluna desejada
IFS	<delimitador>; read -r -a <array>	divide uma dada *string* por um delimitador e retorna um *array* na variável desejada
awk	'{print $<coluna>}'	faz output de uma determinada coluna
sed	's/<char>/<char>/'	faz a substituição de caracteres
head	-n <quantidade>	faz output de todas menos a quantidades de linhas especificadas
sort	-u	faz a ordenação sem repetir entradas
10#<number>		faz com que um dado número seja de base 10
[! -z <string>]		verifica se uma dada *string* é diferente de vazia