

# Sistemas Operativos

## Estatísticas de utilizadores em bash

Professor: Nuno Lau

Duarte Mortágua

91963

Margarida Martins

93169

Universidade de Aveiro

29 de novembro de 2019

# Índice

<b>Introdução</b>	<b>3</b>
<b>Solução e Implementação</b>	<b>4</b>
userstats.sh	4
Argumentos	4
Função data()	4
Sorting	5
comparestats.sh	5
Argumentos	5
Comparação dos ficheiros	5
Sorting	6
<b>Testes</b>	<b>6</b>
userstats.sh	6
Teste das funcionalidades	6
Teste das exceções	8
comparestats.sh	8
Teste das funcionalidades	8
Teste das exceções	9

# Introdução

Este relatório divide-se em três partes:

- Introdução
- Solução e implementação
- Testes de verificação

Na Solução e implementação trataremos de explicar a organização da nossa solução, auxiliando-nos de excertos de código utilizados. Esta explicação tem como objetivo proporcionar a compreensão modular dos scripts. A qualquer momento podemos referir-nos a algo referido anteriormente.

Na parte Testes de verificação iremos mostrar exemplos de testes utilizados ao longo do desenvolvimento dos scripts, `comparestats.sh` e `userstats.sh`.

# Solução e Implementação

## userstats.sh

### Argumentos

O script userstats.sh recebe os seguintes argumentos:

- seleção de utilizadores: -g (grupo) ou -u (utilizador)
- seleção do período: -s (data início) e -e (data fim)
- seleção de ficheiro a tratar: -f <filename>
- ordenação da visualização: -r (decrescente), -n (número de sessões), -t (tempo total), -a (tempo máximo), -i (tempo mínimo).

Para filtrar o funcionamento do script de acordo com os argumentos introduzidos, foram utilizadas flags.

Ou seja, através da função getopt e de um switch case para percorrer os argumentos introduzidos, foram ativadas flags booleanas, indicativas do tipo de seleção e/ou ordenação a fazer, dada pelo utilizador.

Conflitos de flags estão abrangidos.. Um exemplo disto é a inserção de apenas um argumento de ordenação (com a exceção da opção de ordem decrescente -r ), isto é, o utilizador não deve introduzir, por exemplo, os argumentos -a e -t em conjunto. Se tal acontecer, o programa emite uma mensagem de erro e termina.

Todos os outros casos de possíveis conflitos de flags são resolvidos de uma forma similar à acima mencionada. Casos estes que são:

- Inserção de vários argumentos de ordenação;
- Inserção de vários argumentos de seleção de utilizadores;
- Seleção do período de visualização apenas com um delimitador de tempo (-s ou -e);

Uma vez atribuídos os corretos valores booleanos às flags e evitados quaisquer conflitos, o script entra na função data(), que irá gerar o output desejado pelo utilizador.

### Função data()

A primeira coisa a analisar é se o utilizador passou um ficheiro através do argumento -f ou não. Se houver ficheiro introduzido, é corrido o comando last com esse ficheiro como argumento, senão será corrido o comando last apenas, que tem associado o ficheiro wtmp “padrão”. No final, a variável data vai conter todos os utilizadores (únicos) para os quais se vão calcular de seguida as estatísticas.

O cálculo das estatísticas é feito da seguinte forma para cada utilizador:

- É verificado se o utilizador passou argumentos -u ou -g. Em caso afirmativo, é feita uma filtragem de utilizadores através de grep user para -u e através do comando id -G -n user.
- São retirados os valores de tempo de início e fim de sessão, de modo a calcular o tempo máximo, tempo mínimo e tempo total, assim como a ocorrência do user que corresponde ao número de sessões.
- É verificado se foram passados os argumentos de seleção de período. Em caso positivo, as estatísticas são feitas apenas para as datas compreendidas entre as datas inseridas.
- São ignoradas exceções tais como sessões no estado begins, still, gone e reboot.
- No final do cálculo das estatísticas para um user, este é adicionado a uma array.
- Quando os users forem todos percorridos, é impresso o conteúdo do array, contendo todas as estatísticas.

## Sorting

A fase final da função data, depois de obtidas as estatísticas, é saber a forma de imprimir o conteúdo do array. Para isto são avaliadas as flags que foram ativas no tratamento dos argumentos passados pelo utilizador, ao correr o programa.

Se o utilizador introduziu algum argumento de sorting, é ainda verificado se o utilizador ativou a flag reverse order ou não, sendo que depois o código entra na secção sorting correspondente. O comando sort -k 2nr, por exemplo, ordena os valores impressos através da key (-k) coluna número 2 (2n) em modo reverse (r).

## comparestats.sh

### Argumentos

Os argumentos são os dois ficheiros a comparar e as eventuais opções de sorting. É verificado se o utilizador introduziu o número correto de ficheiros, sendo que em caso negativo o programa emite uma mensagem de erro e termina.

Se o utilizador passar um argumento de sorting, este e as respetivas flags são tratados da mesma forma que no userstats.sh, explicado acima.

### Comparação dos ficheiros

Inicialmente é feita a verificação de que os ficheiros introduzidos pelo utilizador existem. Se sim, são criados dois arrays, onde são respetivamente colocadas as linhas dos dois ficheiros introduzidos, de modo a poderem ser trabalhadas pelo script.

De seguida, são percorridas todas as combinações de linhas de ambos os ficheiros, verificando se existem utilizadores iguais em ambos os ficheiros. Se sim, é ativa uma flag “has” que representa “o utilizador existe em ambos os ficheiros” e são computados para variáveis as diferenças dos valores respetivos. Posteriormente, o código passa pela verificação da flag acima referida. Isto é, se o utilizador existir, as variáveis computadas

referidas vão constituir a nova informação do utilizador tratado, sendo esta informação guardada num array final de utilizadores. Caso o utilizador não exista em ambos os ficheiros (se a flag não foi ativa), este é apenas copiado do primeiro ficheiro para a array final de utilizadores, não sofrendo qualquer alteração. De seguida é feito, de forma análoga à anteriormente enunciada, a cópia dos utilizadores presentes no segundo ficheiro que não existem no primeiro, sem qualquer alteração.

## Sorting

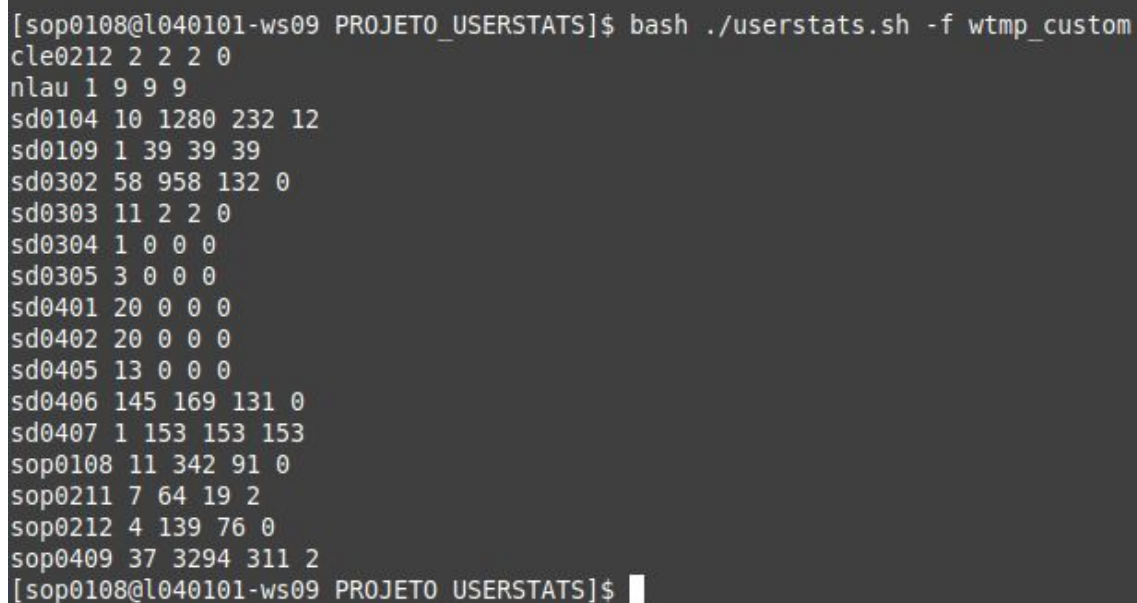
O sorting é feito de forma completamente análoga à do `userstats.sh`, verificando-se as flags ativas, percorrendo o array final de utilizadores e imprimindo-o e fazendo-se o respetivo sorting de seguida.

# Testes

## userstats.sh

### Teste das funcionalidades

Foram testadas as várias opções de escolha.



```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./userstats.sh -f wtmp_custom
cle0212 2 2 2 0
nlau 1 9 9 9
sd0104 10 1280 232 12
sd0109 1 39 39 39
sd0302 58 958 132 0
sd0303 11 2 2 0
sd0304 1 0 0 0
sd0305 3 0 0 0
sd0401 20 0 0 0
sd0402 20 0 0 0
sd0405 13 0 0 0
sd0406 145 169 131 0
sd0407 1 153 153 153
sop0108 11 342 91 0
sop0211 7 64 19 2
sop0212 4 139 76 0
sop0409 37 3294 311 2
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 1- Teste da opção `-f` (usar o ficheiro passado como argumento para a informação sobre as sessões)

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./userstats.sh -g sop
sop0102 1 0 0 0
sop0104 1 1 1 1
sop0106 1 0 0 0
sop0108 14 647 165 0
sop0211 7 64 19 2
sop0212 4 139 76 0
sop0409 67 6085 364 0
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 2- Teste da opção -g (seleção por grupo )

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./userstats.sh -u "s.*"
sd0104 10 1280 232 12
sd0109 1 39 39 39
sd0302 58 958 132 0
sd0303 11 2 2 0
sd0304 1 0 0 0
sd0305 3 0 0 0
sd0401 20 0 0 0
sd0402 20 0 0 0
sd0405 13 0 0 0
sd0406 145 169 131 0
sd0407 1 153 153 153
sop0102 1 0 0 0
sop0104 1 1 1 1
sop0106 1 0 0 0
sop0108 14 647 165 0
sop0211 7 64 19 2
sop0212 4 139 76 0
sop0409 67 6085 364 0
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 3 -Teste da opção -u (seleção por nome de utilizador)

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./userstats.sh -a
sd0304 1 0 0 0
sd0305 3 0 0 0
sd0401 20 0 0 0
sd0402 20 0 0 0
sd0405 13 0 0 0
sop0102 1 0 0 0
sop0106 1 0 0 0
sop0104 1 1 1 1
cle0212 2 2 2 0
sd0303 11 2 2 0
nlau 1 9 9 9
sop0211 7 64 19 2
sd0109 1 39 39 39
sop0212 4 139 76 0
sd0406 145 169 131 0
sd0302 58 958 132 0
sd0407 1 153 153 153
sop0108 14 647 165 0
sd0104 10 1280 232 12
sop0409 67 6085 364 0
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 4- Teste da opção -a (ordenação por tempo máximo)

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./userstats.sh -s "Sep 19 10:00"
-e "Sep 23 18:00"
sop0108 7 154 50 0
sop0409 2 192 156 36
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 5- Teste das opções -s e -e seleção por período)

## Teste das exceções

Foram testadas algumas situações de uso indevido das opções por parte do utilizador.

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./userstats.sh -t -a
Too much sort arguments.
```

Figura 6- Teste das exceções das opções de ordenação

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./userstats.sh -g sop -u "nlau"
Can't sort by group and user at the same time.
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 7- Teste das exceções das opções de seleção dos utilizadores

## comparestats.sh

### Teste das funcionalidades

Foi testado o funcionamento do script a partir dos ficheiros teste1.txt e teste.txt.

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ cat teste1.txt
guida 37 44 29 32
sop1234 88 23 248 91
duarte 123 32 53 18
sop5678 10 3 52 2
sop9012 13 12 44 23
```

Figura 8 - Ficheiro de teste teste1.txt

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ cat teste.txt
sop123 30 2 6 47
sop456 27 29 32 34
guida 32 44 23 12
duarte 100 32 43 53
```

Figura 9- Ficheiro de teste teste.txt



```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./comparestats.sh testel.txt teste.txt -r -t
sop456 27 29 32 34
sop1234 88 23 248 91
sop9012 13 12 44 23
sop5678 10 3 52 2
sop123 30 2 6 47
duarte 23 0 10 -35
guida 5 0 6 20
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 10- Teste da funcionalidade do script

## Teste das exceções

Os testes relacionados com as exceções de sort são idênticos aos acima referidos para o userstats.sh e por isso não estão mencionados aqui. Foram testadas as exceções relacionadas com os ficheiros passados como argumentos.

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./comparestats.sh testel.txt notfound.txt
Error: Files do not exist
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 11- Teste da exceção de nome de ficheiro inválido

```
[sop0108@l040101-ws09 PROJETO_USERSTATS]$ bash ./comparestats.sh testel.txt teste.txt teste3.txt
Wrog number of files
[sop0108@l040101-ws09 PROJETO_USERSTATS]$
```

Figura 12- Teste da exceção de número de ficheiros errado