Sistemas Operativos

Estatísticas de utilizadores em bash

Licenciatura em Engenharia Informática

Pedro Miguel Bastos Almeida – 93150

Primeiro Script – userstats.sh

A estratégia para este script passa, basicamente, por utilizar a função 'getopts' da bash para colocar numa variável todas as alterações ao comando 'last' que o utilizador decide fazer. Depois, para essas opções, coloca-se num array todos os utilizadores que satisfazem as opções. Percorre-se os utilizadores e, para cada um, calcula-se o número de sessões, tempo total, tempo máximo e tempo mínimo de cada um. Finalmente, adiciona-se a variável que contém a ordenação desejada pelo utilizador.

```
#!/bin/bash
sort="sort ";
verificacao=0
options="last"
argumentos=("$@")

usage() { echo "
    Utilização do script:

    ./userstats.sh <options> : calcula os valores de tempo totais, máximos e mínimos das sessões dos utilizadores selecionados

    Options:

    -f <filename> : excuta o script para outro ficheiro

    User fiters:
        -u <name> : filtra os nomes apenas executando o script para o nome especificado

    Date filters:
        -s <Month Day Hour> : excuta o script para datas posteriores à especificada
        -e <Month Day Hour> : excuta o script para datas anteriores à especificada

    Ordering:
        -r : Ordena por ordem decrescente
        -n : Ordena por número de sessões
        -t : Ordena por tempo total das sessões
        -a : Ordena por tempo máximo de sessão
        -i : Ordena por tempo mínimo de sessão

    Nota: Não pode utilizar duas ordenações ao mesmo tempo, a não ser que uma delas seja '-r'


            "
        }
```

Inicialização de variáveis:

- sort : vai servir para as opções de ordenação.
- verificacao : vai servir como flag das ordenações
- options : inicia a variável que vai conter todas as opções colocadas
- Argumentos=("@") vai guardar os argumentos colocados
- usage() : mensagem que explica ao utilizador a utilização do script sempre que o mesmo coloca opções inválidas ou não coloca argumentos necessários

```
while getopts "g:u:s:e:f::nrtai" o; do
    case "${o}" in
        u)
            u="${OPTARG}"
            ;;
        f)
            f=${OPTARG}
            if [ ! -e $f ]; then #se o ficheiro não existe
                echo "O ficheiro não existe, por favor coloque um ficheiro válido!"
                exit 1;
            else
                options="${options} -f $f"
            fi
            ;;
        s)
            if ! date -d "$OPTARG" "+%Y-%m-%d" >/dev/null 2>&1; then #se nao for colocada a data no formato certo
                echo "Please type the date in the format 'month day hour' "
                exit 1;
            fi
            month=$(date -d "$OPTARG" +"%m") #converte a string colocada do mes no respetivo número
            day=${OPTARG:4:2}
            hour=${OPTARG:6}
            if [[ $hour == *":"* ]];then # verifica se a data tem hora (vendo se tem o caracter ':')
                temp=$(date -d "$OPTARG" +"%Y-%m-%d%H:%M")
                options="${options} -s $temp"
            else # não tendo hora, executa com o formato normal sem hora
                options="${options} -s 2019-$month-$day"
            fi
            ;;
        e)
            if ! date -d "$OPTARG" "+%Y-%m-%d" >/dev/null 2>&1; then #se nao for colocada a data no formato certo
                echo "Please type the date in the format 'month day hour' "
                exit 1;
            fi
            month=$(date -d "$OPTARG" +"%m") #converte a string colocada do mes no respetivo número
            day=${OPTARG:4:2}
            hour=${OPTARG:6}
            if [[ $hour == *":"* ]];then # verifica se a data tem hora (vendo se tem o caracter ':'
                temp=$(date -d "$OPTARG" +"%Y-%m-%d%H:%M")
                options="${options} -t $temp"
            else # não tendo hora, executa com o formato normal sem hora
                options="${options} -t 2019-$month-$day"
            fi
            ;;
        n)
            if [ $verificacao == 0 ]; then
                sort+="-n -k2 ";
                verificacao=1
            else #tendo a verificação diferente de 0, já foi usado uma opção de ordenação e por isso dá erro e termina
                echo "Não pode usar duas opções de ordenação!"
                exit 1;
            fi
            ;;
```

```
r)
            sort+="-r";
        ;;
    t)
        if [ $verificacao == 0 ]; then
            sort+="-n -k3 ";
            verificacao=1
        else #tendo a verificação diferente de 0, já foi usado uma opção de ordenação e por isso dá erro e termina
            echo "Não pode usar duas opções de ordenação!"
            exit 1;
        fi
        ;;
    a)
        if [ $verificacao == 0 ]; then
            sort+="-n -k4 ";
            verificacao=1
        else #tendo a verificação diferente de 0, já foi usado uma opção de ordenação e por isso dá erro e termina
            echo "Não pode usar duas opções de ordenação!"
            exit 1;
        fi
        ;;
    i)
        if [ $verificacao == 0 ]; then
            sort+="-n -k5 ";
            verificacao=1
        else #tendo a verificação diferente de 0, já foi usado uma opção de ordenação e por isso dá erro e termina
            echo "Não pode usar duas opções de ordenação!"
            exit 1;
        fi
        ;;
    ?) #colocando uma opção inválida ou não colocando um argumento onde era obrigatório, dá print da mensagem 'usage' e termina
        usage
        exit 1;
        ;;
  esac
done
shift $((OPTIND-1))
```

Getopts da bash:

- -u : guarda numa variável o argumento colocado para depois mais tarde fazer ‘grep’
- -f : guarda o ficheiro passado, se ‘! -e $f’ der verdadeiro, o ficheiro não existe, logo dá erro e termina. Se for falso, adiciona às opções.
- -s, -e : verifica se o formato colocado é o certo, se sim, transforma o mês escrito em número (por exemplo, transforma ‘Nov’ em 11) e guarda o dia e a hora. Depois verifica se a data colocada contém ‘:’ e, se sim, é porque o utilizador colocou hora e adiciona às opções a data com hora. Se não, adiciona também às opções mas sem a hora. No caso do -s, é a data a partir do qual verificamos os utilizadores. Já no caso do -e, é a data até onde queremos verificar os mesmos.
- -n, -t, -a, -i : A variável verificacao certifica-se que só uma destas opções é utilizada, dando erro e terminando se colocarem duas. Ambas adicionam à variável ‘sort’ a coluna por onde é que vai ser ordenado o resultado. Os utilizadores serão ordenados por número de sessões, tempo total, tempo máximo ou tempo mínimo, respetivamente.
- -? : Qualquer outra opção colocada que seja inválida, o programa dá print no terminal da variável ‘usage’ que explica corretamente a sua utilização. Isto também acontece quando o utilizador não coloca argumento em opções com argumento obrigatório.

```
coluna='$1'
pickColuna="| awk '{print $coluna}'"
if [[ ! -z $u ]];then #se foi usado -u, filtra-se já os utilizadores passados
   finalOptions="${options} $pickColuna | grep $u | grep -v "reboot" | grep -v "shutdown" | grep -v "wtmp" | sort | uniq"
else
   finalOptions="${options} $pickColuna | grep -v "reboot" | grep -v "shutdown" | grep -v "wtmp" | sort | uniq"
fi

finalUsers=($(eval $finalOptions))
```

Este trecho de código vai retornar um array dos utilizadores de acordo com as opções colocadas anteriormente. As duas primeiras linhas são apenas para corrigir um bug, pois colocando diretamente '| awk '{print $1}'' na variável 'finalOptions' dava erro. A seguir, verifico se a opção -u foi utilizada e, se sim, faço o 'grep $u' para filtrar os utilizadores e dar apenas aqueles que correspondem ao nome colocado. Se não, retiro o grep e fica só as opções colocadas no 'finalOptions'. A ultima linha serve para ir buscar os utilizadores pretendidos, colocados no 'finalUsers'.

```
getUsersStats() { #função que retorna

if  [[ ${finalUsers[@]} ]] ;then #se existirem utilizadores com as opções selecionadas
   for i in ${finalUsers[@]} #percorre os utilizadores
   do
      if [[ ${argumentos[@]} ]]; then
         for x in ${argumentos[@]} #percorre as opções selecionadas
         do
            if [ $x == "-s" ] || [ $x == "-e" ] || [ $x == "-f" ]  #quando são usadas as opções -s, -e e -f
            then
               counter=($($options | awk '{print $1}' | grep $i | wc -l ))
               timeInicio=($($options | grep "$i" | awk '{print $10}' | grep -v "in" | grep -v "logged" | grep -v "no"))
               break
            else #quando são usadas apenas opções de ordenação
               counter=($( last | awk '{print $1;}' | grep "$i" | wc -l ))
               timeInicio=($( last | grep "$i" | awk '{print $10;}' | grep -v "in" | grep -v "logged" | grep -v "no"))
            fi
         done
      else #se não são usadas opções nenhumas
         counter=($($options | awk '{print $1}' | grep -i "$i" | wc -l ))
         timeInicio=($( $options | grep "$i" | awk '{print $10;}' | grep -v "in" | grep -v "logged" | grep -v "no"))
      fi
```

Entrando agora na função getUsersStats (1ª parte):
Verifica-se se há utilizadores. Se sim, percorremos os utilizadores com um for loop. Para cada utilizador, verificamos se foram utilizadas algumas opções (verificando se a variável 'argumentos' está vazia ou não) e, se sim, percorremos as opções utilizadas. Se forem utilizadas as opções '-s', '-e' ou '-f' o contador do número de sessões ('counter') e o tempo de cada sessão ('timeInicio') vão ter em conta que ficheiro usar e que datas considerar, logo necessita da variável 'options'. Se não for nenhuma dessas opções, significa que são opções de ordenação, que só vão ser aplicadas no final pois não afetam em nada as sessões, sendo assim apenas necessário o comando last sem opções. Finalmente, se não houver opções selecionadas, executa-se apenas o comando last (contido na variável 'options').

```
tempoTotal=0;
        #inicializa as variáveis
        min_time=10000000
        max_time=0
        for k in ${timeInicio[@]} #percorre os tempos de sessão do utilizador
        do
            if [[ ${#k} == 10 ]];then #quando o tempo de sessão está entre 10 e 100 dias
                k0=${k:1:2}
                k1=${k:4:2}
                k2=${k:7:8}
                k2=${k2//}}
                temp_time=$(( 10#$k0*24*60 + 10#$k1*60 + 10#$k2))
            elif [[ ${#k} == 9 ]];then  #quando o tempo de sessão está entre 1 e 10 dias
                k0=${k:1:1}
                k1=${k:3:2}
                k2=${k:6:7}
                k2=${k2//}}
                temp_time=$(( 10#$k0*24*60 + 10#$k1*60 + 10#$k2))
            else #quando o tempo de sessão é inferior a 1 dia
                k1=${k:1:2}
                k2=${k:4:5}
                k2=${k2//}}
                temp_time=$(( 10#$k1*60 + 10#$k2 ))
            fi
            if [[ $temp_time -lt $min_time ]];then #se o tempo calculado for inferior ao tempo mínimo atual, substitui o tempo minimo
                min_time=$temp_time;
            fi
            if [[ $temp_time -gt $max_time ]];then #se o tempo calculado for superior ao tempo maximo atual, substitui o tempo maximo
                max_time=$temp_time;
            fi
            tempoTotal=$(( $tempoTotal+$temp_time )) #adiciona o tempo ao tempo total do utilizador
        done

        userstats+=("$i $counter $tempoTotal $max_time $min_time") #adiciona as stats do utilizador corrente ao array final

    done
else #se não foram encontrados utilizadores
    echo "Não foram encontrados utilizadores para as opções selecionadas!"
fi
}
```

Função getUsersStats (2ª parte):
Para cada user, vamos agora calcular os tempos. Inicializam-se o 'min_time' e o 'max_time' com valores 1000000(apenas precisa de ser um número muito grande) e 0. O uso do '10#' é feito para corrigir um erro de tempos inferiores a 10, p.e 09, a bash considerar base decimal em vez de base 8. Depois, para cada sessão (for loop), verifica-se se o tempo é entre 100 e 10 dias, 10 e 1 dia ou inferior a um dia, através da length do tempo. Entrando num dos 3, calcula-se o tempo e coloca-se na variável 'temp_time' que vai guardar temporariamente o tempo de sessão. A seguir, verifica-se se o tempo é menor que o mínimo atual ou maior que o máximo atual e, se for, substitui-se. No fim, adiciona-se o tempo da sessão ao tempo total. Já fora do for loop, adiciona-se ao array 'userstats' as "stats" do utilizador, ou seja, "nome($i) contador de sessões($counter) tempo total($tempoTotal) tempo máximo($max_time) e tempo mínimo($min_time)". No fim, caso a primeira verificação desta função der falso, significa que não encontrou utilizadores e, portanto, dá print no terminal que não encontrou utilizadores.

```
getUsersStats #calcula as stats dosutilizadores
printf "%s\n" "${userstats[@]}" | ${sort} #print das stats dos utilizadores com a ordenação (variavel sort) selecionada
```

Finalmente, chama-se a função 'getUsersStats' que calcula as estatísticas dos utilizadores selecionados e coloca-os na variável 'userstats'. Depois utiliza-se o 'printf' para dar print de um utilizador por linha, não esquecendo de adicionar a variável 'sort' que os ordena de acordo com o que o utilizador escolheu.

Segundo Script – comparestats.sh

Para o segundo script, a estratégia utilizada é simples. Faz se o mesmo tratamento das opções de ordenação utilizado no script anterior. Depois, lê-se linha a linha cada ficheiro e guarda-se o conteúdo em variáveis. Percorre-se as 'stats' do primeiro ficheiro e depois para cada um, percorre se as 'stats' do segundo ficheiro (2 for loops). Verifica-se, para cada utilizador do primeiro ficheiro, se o mesmo está contido no segundo. Se sim, subtrai-se os valores do primeiro aos do segundo ficheiro e adiciona-se ao array 'finalStats'. Por fim, percorre-se outra vez os utilizadores do primeiro ficheiro e, os utilizadores que não estiverem no 'finalStats', são adicionados. O mesmo acontece para o segundo ficheiro. Ou seja, seum utilizador está em ambos os ficheiros, é calculada a diferença dos seus valores. Se apenas estiver num dos ficheiros, é copiado.

Parte inicial das opções:
Visto que as opções são as opções de ordenação (sort) usadas no script anterior, não vai estar explicado aqui essa parte, pois o código é o mesmo e já está especificado em cima.

```
if [[ ! -e $1 ]];then
    echo "Primeiro ficheiro não encontrado!"
    exit 1;
else
    userstats1=$1
fi

if [[ ! -e $2 ]];then
    echo "Segundo ficheiro não encontrado!"
    exit 1;
else
    userstats2=$2
fi
```

Esta parte do código apenas serve para verificar se os ficheiros passados como primeiro e segundo argumento existem. Guarda-se nas variáveis 'userstats1' e 'userstats2' o primeiro e segundo ficheiro, respetivamente.

```
while IFS= read -r linha
do
    IFS=$'\n'
  statsFirstUser+=($linha);
done < $userstats1

while IFS= read -r linha
do
    IFS=$'\n'
  statsSecondUser+=($linha);
done < $userstats2
```

Aqui, guarda-se nas variáveis 'statsFirstUser' e 'statsSecondUser' o conteúdo do primeiro e segundo ficheiro, respetivamente.

```
for user1 in ${statsFirstUser[@]}
do
   nome1=$(echo "$user1" | awk '{print $1}');
   for user2 in ${statsSecondUser[@]}
   do
      nome2=$(echo "$user2" | awk '{print $1}');
      if [ $nome1 == $nome2 ]; then
         sessoes=$(($(echo "$user1" | awk '{print $2}') - $(echo "$user2" | awk '{print $2}')))
         tempoTotal=$(($(echo "$user1" | awk '{print $3}') - $(echo "$user2" | awk '{print $3}')))
         minTemp=$(($(echo "$user1" | awk '{print $4}') - $(echo "$user2" | awk '{print $4}')))
         maxTemp=$(($(echo "$user1" | awk '{print $5}') - $(echo "$user2" | awk '{print $5}')))
         finalStats+=("$nome1 $sessoes $tempoTotal $minTemp $maxTemp")
      fi
   done
done
```

Dois for loops:
Percorre-se os utilizadores do primeiro ficheiro e, para cada um, percorre-se os do segundo. Se '$nome1==$nome2', então esse utilizador está em ambos os ficheiros, pelo que valos subtrair os seus valores. Em todas as variáveis é feita a subtração da mesma maneira, com exceção da coluna selecionada, que pode ser $2, $3, $4 e $5. Finalmente adiciona-se ao array final 'finalStats' o utilizador já com as diferenças dos valores.

```
for user1 in ${statsFirstUser[@]}
do
   flag=0
   nome1=$(echo "$user1" | awk '{print $1}');
   for finalUser in ${finalStats[@]}
   do
      nomeFinal=$(echo "$finalUser" | awk '{print $1}');
      if [ $nome1 == $nomeFinal ] ;then
         flag=1;
      fi
   done

   if [[ $flag -eq 0 ]]; then
      finalStats+=("$user1")
   fi
done

for user2 in ${statsSecondUser[@]}
do
   flag=0
   nome2=$(echo "$user2" | awk '{print $1}');
   for finalUser in ${finalStats[@]}
   do
      nomeFinal=$(echo "$finalUser" | awk '{print $1}');
      if [ $nome2 == $nomeFinal ] ;then
         flag=1;
      fi
   done

   if [[ $flag -eq 0 ]]; then
      finalStats+=("$user2")
   fi
done
```

Depois de tratar os utilizadores que estão em ambos os ficheiros, vamos encontrar os utilizadores que só estão num deles. Para isto, percorremos os utilizadores do primeiro ficheiro, e para cada um deles percorremos os utilizadores do array final. Se não estiverem no array final,a variável 'flag' fica a 0, significando que só está no primeiro ficheiro. Assim, copia-se os seus valores para o array final. O mesmo acontece para o segundo ficheiro.

```
IFS=$' ' #para conseguir dar print
printf '%s \n' "${finalStats[@]}" | ${sort}
```

Finalmente, dá-se print dos utilizadores calculados, voltando a não esquecer a variável 'sort' que os ordena de acordo com o que o utilizador escolheu.

Testes Realizados

Depois da realização dos scripts, foram feitos alguns testes à solução final.

Para o primeiro script:

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh
bastos 13 3022 1083 0
```

A partir daqui todos os testes foram realizados ao ficheiro 'wtmp' (incluído na pasta submetida) pois contém vários utilizadores, essencial para a verificação da solução.

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh  -f wtmp
nlau 6 11 4 0
sd0104 4 579 230 78
sd0105 10 129 44 0
sd0106 1 0 0 0
sd0109 2 144 131 13
sd0301 60 0 0 0
sd0302 256 1399 133 0
sd0303 28 4 2 0
sd0304 1 0 0 0
sd0305 20 0 0 0
sd0401 21 0 0 0
sd0402 90 133 131 0
sd0403 1 9 9 9
sd0405 610 46 10 0
sd0406 182 354 154 0
sd0407 2 186 136 50
sop0101 14 1614 1314 0
sop0106 1 0 0 0
sop0202 17 5487 4410 3
sop0301 3 17111 8486 139
sop0402 18 2851 1439 35
sop0406 10 305 165 0
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh  -f wtmp -u "sop.*"
sop0101 14 1614 1314 0
sop0106 1 0 0 0
sop0202 17 5487 4410 3
sop0301 3 17111 8486 139
sop0402 18 2851 1439 35
sop0406 10 305 165 0
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh  -f wtmp -u "sop.*" -s "Nov 13 20:00" -e "Nov 15 20:00"
sop0101 2 106 102 4
sop0106 1 0 0 0
sop0402 3 213 110 35
sop0406 10 305 165 0
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh  -f wtmp -t -u "sop.*"
sop0106 1 0 0 0
sop0406 10 305 165 0
sop0101 14 1614 1314 0
sop0402 18 2851 1439 35
sop0202 17 5487 4410 3
sop0301 3 17111 8486 139
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh  -f wtmp -n -u "sop.*"
sop0106 1 0 0 0
sop0301 3 17111 8486 139
sop0406 10 305 165 0
sop0101 14 1614 1314 0
sop0202 17 5487 4410 3
sop0402 18 2851 1439 35
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh  -f wtmp -t -r -u "sop.*
sop0301 3 17111 8486 139
sop0202 17 5487 4410 3
sop0402 18 2851 1439 35
sop0101 14 1614 1314 0
sop0406 10 305 165 0
sop0106 1 0 0 0
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh  -f wtmp -a -r -u "sop.*"
sop0301 3 17111 8486 139
sop0202 17 5487 4410 3
sop0402 18 2851 1439 35
sop0101 14 1614 1314 0
sop0406 10 305 165 0
sop0106 1 0 0 0
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh  -f wtmp -i -r -u "sop.*"
sop0301 3 17111 8486 139
sop0402 18 2851 1439 35
sop0202 17 5487 4410 3
sop0406 10 305 165 0
sop0106 1 0 0 0
sop0101 14 1614 1314 0
```

Testes de erros:

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh -u bass
Não foram encontrados utilizadores para as opções selecionadas!
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh -u
./userstats.sh: option requires an argument -- u

        Utilização do script:

        ./userstats.sh <options> : calcula os valores de tempo totais, máximos e minimos das sessões dos utilizadores selecionados

        Options:

        -f <filename> : excuta o script para outro ficheiro

        User fiters:
                -u <name> : filtra os nomes apenas executando o script para o nome especificado

        Date filters:
                -s <Month Day Hour> : excuta o script para datas posteriores à especificada
                -e <Month Day Hour> : excuta o script para datas anteriores à especificada

        Ordering:
                -r : Ordena por ordem decrescente
                -n : Ordena por número de sessões
                -t : Ordena por tempo total das sessões
                -a : Ordena por tempo máximo de sessão
                -i : Ordena por tempo mínimo de sessão

        Nota: Não pode utilizar duas ordenações ao mesmo tempo, a não ser que uma delas seja '-r'
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh -y
./userstats.sh: illegal option -- y

        Utilização do script:

        ./userstats.sh <options> : calcula os valores de tempo totais, máximos e minimos das sessões dos utilizadores selecionados

        Options:

        -f <filename> : excuta o script para outro ficheiro

        User fiters:
                -u <name> : filtra os nomes apenas executando o script para o nome especificado

        Date filters:
                -s <Month Day Hour> : excuta o script para datas posteriores à especificada
                -e <Month Day Hour> : excuta o script para datas anteriores à especificada

        Ordering:
                -r : Ordena por ordem decrescente
                -n : Ordena por número de sessões
                -t : Ordena por tempo total das sessões
                -a : Ordena por tempo máximo de sessão
                -i : Ordena por tempo mínimo de sessão

        Nota: Não pode utilizar duas ordenações ao mesmo tempo, a não ser que uma delas seja '-r'
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh -f wt
O ficheiro não existe, por favor coloque um ficheiro válido!
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh -s "Nov 2320"
Please type the date in the format 'month day hour'
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./userstats.sh -t -i
Não pode usar duas opções de ordenação!
```

Para o segundo script:

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./comparestats.sh userstats_1 userstats_2
nlau 2 2 2 0
sd0104 2 379 30 7
sd0105 7 52 24 0
sd0106 0 0 0 0
sd0109 0 131 122 11
sd0301 54 0 0 0
sd0302 6 1260 120 0
sd0303 6 2 1 0
sd0304 0 0 0 0
sd0305 3 0 0 0
sd0401 7 0 0 0
sd0402 20 33 41 0
sd0403 0 5 5 5
sd0405 210 3 1 0
sd0406 0 0 0 0
sd0407 0 46 6 41
sop0101 0 1453 1301 0
sop0106 0 0 0 0
sop0202 0 5433 4366 2
sop0301 0 15400 7638 126
sop0402 0 2566 1296 32
sop0406 0 0 0 0
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./comparestats.sh -r userstats_1 userstats_2
sop0406 0 0 0 0
sop0402 0 2566 1296 32
sop0301 0 15400 7638 126
sop0202 0 5433 4366 2
sop0106 0 0 0 0
sop0101 0 1453 1301 0
sd0407 0 46 6 41
sd0406 0 0 0 0
sd0405 210 3 1 0
sd0403 0 5 5 5
sd0402 20 33 41 0
sd0401 7 0 0 0
sd0305 3 0 0 0
sd0304 0 0 0 0
sd0303 6 2 1 0
sd0302 6 1260 120 0
sd0301 54 0 0 0
sd0109 0 131 122 11
sd0106 0 0 0 0
sd0105 7 52 24 0
sd0104 2 379 30 7
nlau 2 2 2 0
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./comparestats.sh -t userstats_1 userstats_2
sd0106 0 0 0 0
sd0301 54 0 0 0
sd0304 0 0 0 0
sd0305 3 0 0 0
sd0401 7 0 0 0
sd0406 0 0 0 0
sop0106 0 0 0 0
sop0406 0 0 0 0
nlau 2 2 2 0
sd0303 6 2 1 0
sd0405 210 3 1 0
sd0403 0 5 5 5
sd0402 20 33 41 0
sd0407 0 46 6 41
sd0105 7 52 24 0
sd0109 0 131 122 11
sd0104 2 379 30 7
sd0302 6 1260 120 0
sop0101 0 1453 1301 0
sop0402 0 2566 1296 32
sop0202 0 5433 4366 2
sop0301 0 15400 7638 126
```

Testes de erros:

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./comparestats.sh -y userstats_1 userstats_2
./comparestats.sh: illegal option -- y

        Utilização do script:

        ./comparestats.sh <options> <filename1> <filename2> : compara os dois ficheiros mostrando a diferença dos valores

        Options:

        Ordering:
                -r : ordena por ordem decrescente
                -n : ordena pelo número de sessões
                -t : ordena pelo tempo total
                -a : ordena pelo temo máximo
                -i : ordena pelo tempo mínimo
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./comparestats.sh userstats_ userstats_2
Primeiro ficheiro não encontrado!
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./comparestats.sh userstats_1 userstats_
Segundo ficheiro não encontrado!
```

```
bastos@bastos-VirtualBox:~/Downloads/TESTES$ ./comparestats.sh -t -i userstats_1 userstats_2
Não pode usar duas opções de ordenação!
```