universidade de aveiro

Sistemas Operativos

Professor Nuno Lau

Departamento de Eletrónica, Telecomunicações e Informática

Monitorização de interfaces de rede em bash

Ana Raquel Paradinha 102491

Paulo Pinto 103234

A participação na realização do trabalho foi igualitária.

Índice

1 Introdução

No âmbito da unidade curricular Sistemas Operativos foi-nos proposto o desenvolvimento do *script netifstat.sh* em *bash* que permite analisar a quantidade de dados transmitidos e recebidos, assim como as respetivas taxas de transferência, por cada interface de rede ativa na nossa máquina.

Com efeito, com base nos conteúdos explorados nas aulas práticas e alguma pesquisa em fontes externas conseguimos chegar ao resultado esperado. Nesse sentido, no presente relatório descrevemos as metodologias utilizadas para chegar a uma solução, assim como as dificuldades que encontramos e como foram resolvidas.

2 Organização do código

2.1 Declaração de variáveis globais

```
declare -A rx           # Array associativo que guarda os valores de rx
declare -A tx           # Array associativo que guarda os valores de tx       # O index corresponde ao nome da interface
declare -A rrate        # Array associativo que guarda os valores de rrate
declare -A trate        # Array associativo que guarda os valores de trate
declare -A rx_last      # Array associativo que guarda os valores de rx do loop anterior
declare -A tx_last      # Array associativo que guarda os valores de tx do loop anterior
declare -A rx_total     # Array associativo que guarda a soma dos valores de rx a cada loop
declare -A tx_total     # Array associativo que guarda a soma dos valores de tx a cada loop

rexp='^[0-9]+(\.[0-9]*)?$'      # Expressão regex para verificar se o último arg é um número
SEC=${@: -1}                    # Último argumento é o número de segundos
NAME=""                         # Expressão regex para selecionar as interfaces
NUMBER=""                       # Número de interfaces que queremos ver
ctrl=0                          # Valor de controlo para as opções de unidades
ord=0                           # Valor de controlo para as opções de ordenação
exp=0                           # Valor do expoente para fazer a conversão de unidades
loop=0                          # Ativa ou desativa o loop
col=1                           # Identifica a coluna por onde vão ser ordenados os dados
reverse=""                      # Ativa ou desativa o sort reverso
turn=0                          # Indica se é a primeira vez que o loop é executado
```

Figura 1

Para o tratamento de dados, decidimos usar *arrays* associativos uma vez que nos permitem guardar valores associados a uma chave, a qual no nosso caso corresponde ao nome de cada interface de rede analisada.

Assim, definimos os seguintes *arrays*:

- **tx**, **rx** → guardam a quantidade de dados transmitidos e recebidos em cada interface de rede;
- **trate**, **rrate** → guardam as taxas de transferência para os períodos de tempo indicados ao executar o *script*;
- **tx_last**, **rx_last** → são usados quando a opção -l (loop) está ativa para passar os valores da quantidade de dados da segunda leitura do loop anterior para a primeira leitura do loop atual;
- **tx_last**, **rx_last** → guardam a soma cumulativa dos valores de tx e rx a cada loop.

Além disso, definimos uma expressão regular (**rexp**) para fazer a verificação de que o último argumento passado corresponde à quantidade de segundos que se pretende analisar, sendo que esta pode ser um número positivo inteiro ou decimal.

De seguida, inicializamos três variáveis dependentes dos valores passados como argumentos no terminal:

- **SEC** → número de segundos para análise, corresponde ao último argumento;

- **NAME** → corresponde à expressão regular pela qual vão ser filtrados os nomes das interfaces de rede a apresentar, só é definida quando a opção -c é passada como argumento;
- **NUMBER** → indica quantas interfaces vão ser apresentadas, só é definida quando a opção -p é passada como argumento.

Para efeitos de controlo das opções da unidade de visualização (-b, -k, -m) e de ordenação (-t, -r, -T, -R), uma vez que só pode ser passada uma de cada, definimos as variáveis **ctrl** e **ord**, respetivamente. Assim, estas variáveis são inicializadas a 0 (zero) e passam a 1 quando uma das opções que controlam é passada como argumento.

Seguidamente, temos a variável **exp** cujo valor vai ser usado como expoente para calcular a conversão de unidades, podendo ser 0, 1 ou 2 para bytes, kilobytes e megabytes, respetivamente.

Também definimos a variável **loop** que é ativada quando a opção -l é passada ao *script*.

A nível das opções de ordenação temos ainda a variável **col** que identifica a coluna da tabela pela qual vai ser feita a ordenação e variável **reverse** que ativa/desativa o *sort* reverso. Esta última corresponde a uma *null string* quando a opção -v não está ativa e passa para “r” quando é ativada.

Por fim, a variável **turn** é usada quando a opção -l está ativa para indicar se o loop está na primeira iteração (valor 0) ou não (valor 1).

2.2 Tratamento de argumentos

Para o tratamento de argumentos foi utilizado o comando *getops,* que nos permite registar vários argumentos. Se esses argumentos forem válidos e estiverem corretamente aplicados, o comando *getops* vai proceder ao tratamento destes e realizar as tarefas que correspondem aos argumentos.

2.2.1 Função *options()*

```
# Lista as opções disponíveis
options() {
    echo "--------------------------------------------------------------------------------"
    echo "${@:0} -c [NAME] -b|-k|-m -p [NUMBER] -t|-r|-T|-R -v -l [SEC]"
    echo
    echo "OPÇÕES DISPONÍVEIS!"
    echo
    echo "    -c    : Selecionar as interfaces a analisar através de uma expressão regular"
    echo "    -p    : Defenir o número de interfaces a visualizar"
    echo "    -l    : Analisar as interfaces de s em s segundos"
    echo "    -v    : Fazer um sort reverso"
    echo "O último argumento tem de corresponder sempre ao número de segundos que pretende analisar."
    echo
    echo "Opções de unidades (usar apenas 1):"
    echo "    -b    : Valores em bytes (default)"
    echo "    -k    : Valores em Kilobytes"
    echo "    -m    : Valores em Megabytes"
    echo
    echo "Opções de ordenação (usar apenas 1):"
    echo "    -t    : Ordenar pelo TX"
    echo "    -r    : Ordenar pelo RX"
    echo "    -T    : Ordenar pelo TRATE"
    echo "    -R    : Ordenar pelo RRATE"
    echo "A ordenação default é alfabética e para cada opção é decrescente."
    echo "--------------------------------------------------------------------------------"
}
```

Figura 2

A função ***options*** é utilizada para descrever como deve ser executado o *script netifstat.sh*, indicando as opções disponíveis, assim como a sua funcionalidade, e quais destas exigem argumentos.

2.2.2 Função *error_exit()*

```
error_exit () {
    options
    exit 1
}
```

Figura 3

A função ***error_exit*** é chamada sempre que um dos argumentos é usado de forma incorreta. Nesse sentido, executa a função *options* e termina o programa com o comando *exit 1*.

2.2.3 Função *unit_exit()*

```
unit_exit () {
    if [[ $ctrl == 1 ]]; then
        # Quando há mais que 1 argumento de unidades
        echo "ERRO: não é possivel usar -b, -k e -m ao mesmo tempo!" >&2
        error_exit
    else
        ctrl=1
    fi
}
```

Figura 4

No caso dos argumentos -b, -k e/ou -*m* serem usados será chamada a função ***unit_exit*** que averigua se estes não são usados em simultâneo. Fazendo uso da variável de controlo *ctrl,* se esta for igual a 0 significa que é a primeira vez que um deles é usado, permitindo assim continuar a verificação dos restantes argumentos e alterando o seu valor para 1.

Se a variável de controlo for igual a 1 quer dizer que já houve um argumento deste tipo a ser usado, levando a função a apresentar uma mensagem de erro relativa ao mau uso dos argumentos -b, -k e/ou -m seguida da execução da função *error_exit.*

2.2.4 Função *sort_exit()*

```
sort_exit () {
    reverse="r"
    if [[ $ord == 1 ]]; then
        echo "Não é possivel usar -t,-r,-T e -R ao mesmo tempo!" >&2
        error_exit
    else
        ord=1
    fi
}
```

Figura 5

Para o tratamento das opções -t, -r, -T e -R é usada uma fórmula semelhante. Sempre que uma delas é usada a função ***sort_exit*** é chamada definindo o valor de *reverse* como “r” e usando a variável de controlo *ord* verifica se alguma destas opções já foi utilizada antes.

Se for a primeira vez a variável *ord* será definida como 1 e o tratamento de argumentos prossegue, se for a segunda vez, será impressa uma mensagem de erro relativa a estes argumentos e é chamada a função *error_exit*.

2.2.5 *getops*

```
# Verifica que o argumento obrigatório está presente
if [[ $# == 0 ]]; then
    echo "ERRO: deve passar pelo menos um argumento (número de segundos a analisar)." >&2
    error_exit
fi

# Verifica que o último argumento é o número de segundos
if ! [[ $SEC =~ $rexp ]]; then
    echo "ERRO: o último argumento tem de ser o número de segundos que pretende analisar." >&2
    error_exit
fi
```

Figura 6

Antes de usar o comando *getops* são utilizados dois *if* que irão verificar se existe pelo menos um argumento passado e se o último argumento é um valor numérico inteiro ou decimal. Caso uma destas condições falhe é chamada a função *error_exit (Fig.)* e impressa uma mensagem de erro.

```
set -- "${@:1:$(($#-1))}"   #retira o último arg (sec) para não ser usado como arg das opções

# Tratamento das opções passadas como argumentos
while getopts ":c:bkmp:trTRvl" option; do
    case $option in
    c) # Seleção das interfaces a visualizar através de uma expressão regular
        NAME=$OPTARG
        ;;
    b)
        unit_exit     # Unidade = Byte
        ;;
    k)
        unit_exit
        exp=1     # Unidade = KiloByte
        ;;
    m)
        unit_exit
        exp=2     # Unidade = MegaByte
        ;;
    p) # Número de interfaces a visualizar
        NUMBER=$OPTARG
        if [[ NUMBER =~ "^[0-9]+$" ]]; then
            echo "ERRO: o número de interfaces tem de ser um inteiro positivo." >&2
            error_exit
        fi
        ;;
    t)
        sort_exit
        col=2
        ;;
    r)
        sort_exit
        col=3
        ;;
    T)
        sort_exit
        col=4
        ;;
    R)
        sort_exit
        col=5
        ;;
    v) #Ordenação reversa
        if [[ $reverse == "r" ]];  then
            reverse=""
        else
            reverse="r"
        fi
        ;;
    l) # Loop
        loop=1
        ;;
    :) # Argumento obrigatório em falta
        echo "ERRO: argumento em falta na opção -${OPTARG}!" >&2
        error_exit
        ;;
    *) #Passagem de argumentos inválidos
        echo "ERRO: opção inválida!" >&2
        error_exit
        ;;
    esac
done
```

Figura 7

Inicialmente, retiramos o último argumento, correspondente ao número de segundos estabelecido, do *array* predefinido *$@*, que agrupa todos os argumentos e

opções passados na linha de comandos, com o objetivo de este não ser associado erradamente ao argumento de uma das opções definidas.

De seguida, usamos o comando *getopts* para tratar cada uma das opções passadas. Ao usar a expressão ":c:bkmp:trTRvl" definimos que o *getops* vai correr em modo *silent error checking* através do primeiro caráter ":", o que nos permite definir o tratamento de erros que pretendermos.

Nesse sentido, quando a opção **-c** é verificada atribuímos o argumento (obrigatório) que lhe segue à variável *NAME*.

No caso das opções **-b**, **-k** e **-m** para além de ser executada a função *unit_exit* é definido o valor de *exp* como 1 para -k e 2 para -m. Quando a opção é -b este valor não se altera, pois é o default.

Para a opção **-p** o argumento que lhe segue (obrigatório) é atribuído à variável *NUMBER* e verifica-se se este é um inteiro positivo. No caso de não ser, é reportado um erro.

Adicionalmente, quando temos **-t**, **-r**, **-T** ou **-R**, para além da verificação já descrita (função *sort_exit*) atribuímos os valores 2, 3, 4 e 5 à variável *col*, respetivamente.

Na opção **-v** o valor de *reverse* é alterado para "r" ou " " consoante o seu valor atual. Em **-l** apenas alteramos o valor de *loop* para 1.

Por fim, quando uma opção que exige argumento não o recebe, passa a constituir a variável $OPTARG da opção **":"**, que desencadeará um erro. Além disso, se for passada uma opção não definida é executada a opção **"*"** que relata esse erro.

3 Listagem de processos

```
printData() {
    n=0
    un=$((1024 ** exp))
    for net in /sys/class/net/[[:alnum:]]*; do
        if [[ -r $net/statistics ]]; then
            f="$(basename -- $net)"
            if ! [[ $NAME =~ ""  && $f =~ $NAME ]]; then
                continue
            fi

            if [[ $turn == 0 ]]; then
                rx_bytes1=$(cat $net/statistics/rx_bytes | grep -o -E '[0-9]+') # está em bytes
                tx_bytes1=$(cat $net/statistics/tx_bytes | grep -o -E '[0-9]+') # está em bytes
                sleep $SEC
            else
                rx_bytes1=rx_last[$f]
                tx_bytes1=tx_last[$f]
            fi
```

Figura 8

A função ***printData*** é a principal do programa, pois é responsável por obter os valores necessários e imprimi-los.

Assim, a função começa por definir uma variável local **n** que serve para verificar quando o número de interfaces dado pela opção *-p* é atingido, caso esta seja selecionada.

Outra variável definida é ***un*** que será usada para formatar os valores em bytes, kilobytes ou megabytes dependendo dos argumentos passados.

De seguida é criado um *for* que vai buscar o caminho da pasta de cada interface (por exemplo */sys/class/net/eth0*), no qual ***[[:alnum:]]**** indica que deve procurar por nomes alfanuméricos. O *if* posterior confirma a existência da pasta *statistics* dentro da pasta de cada interface e só se tal acontecer é que o processo continua, caso contrário esta é ignorada e o ciclo *for* continua para a próxima interface.

No caso da pasta *statistics* existir, é definida uma variável *f* com o nome da interface através do comando *basename.* De seguida é criado um *if* que verifica se o nome da interface corresponde à expressão passada no argumento de -c, se tal não for verdade a interface é ignorada e o ciclo for passa para a seguinte. Caso o *-c* não seja definido este if é sempre falso, ou seja, todas as interfaces serão impressas.

O segundo *if* verifica se é a primeira vez que a função *printData* está a ser chamada. Este *if* é útil apenas quando o argumento *-l* é passado, o que leva a função a ocorrer em loop, enquanto que no resto dos casos ela apenas é chamada uma vez.

Para tal decisão é usada a variável de controlo *turn,* se esta for igual a 0 significa que é a primeira vez que a função *printData* é chamada e os primeiros valores de rx e tx são guardados nas variáveis ***rx_bytes1*** e ***tx_bytes1***, respetivamente. O

comando ***cat*** serve para ir buscar o conteúdo da pasta especificada e o comando ***grep -o -E '[0-9]+'*** confirma que apenas sejam guardados valores numéricos.

O *else* da condição só é executado quando a opção *loop* está ativa e tem por objetivo atribuir os valores guardados da análise anterior para serem usados como primeira medição da atual.

```
        rx_bytes2=$(cat $net/statistics/rx_bytes | grep -o -E '[0-9]+') #está em bytes
        tx_bytes2=$(cat $net/statistics/tx_bytes | grep -o -E '[0-9]+') #está em bytes

        rx[$f]=$((rx_bytes2 - rx_bytes1))
        tx[$f]=$((tx_bytes2 - tx_bytes1))

        rrate[$f]=$(bc <<< "scale=3;${rx[$f]}/$SEC")
        trate[$f]=$(bc <<< "scale=3;${tx[$f]}/$SEC")

        if [[ $loop == 1 ]]; then
            tx_total[$f]=$((tx_total[$f] + tx[$f]))
            rx_total[$f]=$((rx_total[$f] + rx[$f]))
            rx_last[$f]=$rx_bytes2
            tx_last[$f]=$tx_bytes2
        fi
        n=$((n + 1))
    fi
    if [[ $n == $NUMBER ]]; then
        break
    fi
done
```

Figura 9

Em continuação, após o intervalo definido, determinamos os segundos valores de rx e tx pelo mesmo processo adotado para *rx_bytes1* e *tx_bytes1*. Com esses dados calculamos a diferença entre ambos para obter o tx e rx de cada interface de rede e adicionamo-los aos respetivos *arrays*.

De seguida, com o auxílio do comando *basic calculator* avaliamos o valor das taxas de transferência, **trate** e **rrate**, dividindo os valores de rx e tx da interface atual pelo intervalo de tempo definido.

O *if* que se segue só é executado quando a opção -l está ativa e nele incrementamos os **tx_total** e **rx_total** com os valores tx e rx da interface analisada em cada iteração do ciclo. Também atribuímos às variáveis ***tx_last*** e ***rx_last*** os valores de *tx_bytes2* e *rx_bytes2*, respetivamente, com o objetivo de ficarem acessíveis na execução seguinte do loop a fim de serem usadas como primeiro valor de tx e rx.

Além disso, a variável **n** é incrementada e comparada ao valor de *NUMBER*, quando a opção -p é passada no terminal, e termina o ciclo *for* quando estes são iguais, pois significa que foi atingido o número pretendido.

```
    n=0
    for net in /sys/class/net/[[:alnum:]]*; do
        if [[ -r $net/statistics ]]; then
            f="$(basename -- $net)"
            if ! [[ $NAME =~ ""  && $f =~ $NAME ]]; then
                continue
            fi
            if [[ $loop == 1 ]]; then
                printf "%-12s %12s %12s %12s %12s %12s %12s\n" "$f" "$(bc <<< "scale=3; ${tx[$f]}/$un")"
                "$(bc <<< "scale=3; ${rx[$f]}/$un")" "$(bc <<< "scale=3;${trate[$f]}/$un")" "$(bc <<< "scale=3;${rrate[$f]}/$un")"
                "$(bc <<< "scale=3;${tx_total[$f]}/$un")" "$(bc <<< "scale=3;${rx_total[$f]}/$un")"
            else
                printf "%-12s %12s %12s %12s %12s\n" "$f" "$(bc <<< "scale=3; ${tx[$f]}/$un")" "$(bc <<< "scale=3; ${rx[$f]}/$un")"
                "$(bc <<< "scale=3;${trate[$f]}/$un")" "$(bc <<< "scale=3;${rrate[$f]}/$un")"
            fi
            n=$((n + 1))
        fi
        if [[ $n == $NUMBER ]]; then
            break
        fi
    done | sort -k$col$reverse
}
```

Figura 10

Seguidamente, o valor de **n** é novamente passado a zero, para ser utilizado num novo ciclo *for*.

As seis linhas de código seguintes são semelhantes à do ciclo anterior, cuja explicação já foi feita.

Nesse sentido, na condição *if* subsequente é executado o primeiro bloco caso a opção -l esteja ativa ou o segundo (*else*) caso contrário. Ambos convertem os dados obtidos às unidades especificadas pelas opções -b, -k ou -m, com recurso, novamente, ao comando bc e à variável *un*, definida anteriormente, e imprimem os mesmos numa tabela.

Os comandos até ao fim do ciclo *for* são uma repetição dos anteriores, sendo, no entanto, necessários para a execução do comando ***sort***. Com efeito, este recebe como argumento **-k** o valor de *col*, referente à coluna pela qual deve fazer a ordenação, e o de *reverse* que indica se será seguida a ordem default (crescente) para reverse igual a “ “ ou se será decrescente, com reverse a tomar o valor “r”.

```
if [[ $loop == 1 ]]; then
    while true; do
        printf "%-12s %12s %12s %12s %12s %12s %12s %12s\n" "NETIF" "TX" "RX" "TRATE" "RRATE" "TXTOT" "RXTOT"
        printData
        turn=1
        echo ""
        sleep $SEC
    done
else
    printf "%-12s %12s %12s %12s %12s\n" "NETIF" "TX" "RX" "TRATE" "RRATE"
    printData
fi
```

Figura 11

A última condição *if*, caso o *loop* esteja ativo, executa um ciclo *while* infinito, que imprime o cabeçalho da tabela com as duas colunas exclusivas desta opção (txtot e rxtot), chama a função *printData*, altera o valor de *turn* para 1, identificando que já foi feito um loop e corre o comando sleep para voltar a realizar uma medição de dados ao fim do tempo definido.

Por fim, caso se pretenda fazer a leitura uma única vez, é executado o bloco *else*, que apenas imprime o cabeçalho da tabela e corre a função *printData*.

4 Resultados

Nesta secção mostramos algumas das possíveis combinações de opções para executar o script *netifstat.sh*, as quais serviram também para verificarmos que eram obtidos os resultados esperados.

Por conseguinte, começamos pelo mais simples, passando apenas o número de segundos a analisar.

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh 10
NETIF                 TX          RX       TRATE       RRATE
lo               172.000     172.000      17.200      17.200
virbr0                 0           0           0           0
virbr0-nic             0           0           0           0
wlp0s20f3      67660.000   37499.000    6766.000    3749.900
```

Figura 12

O resultado esperado era uma tabela com todas as interfaces de rede ativas ordenadas por ordem alfabética, o que se verifica.

Seguidamente, usamos a opção -c com a expressão regular “l”, pelo que se espera que apareçam apenas as interfaces que contêm este caráter no nome.

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -c l 10
NETIF                 TX          RX       TRATE       RRATE
lo                     0           0           0           0
wlp0s20f3      47970.000   31808.000    4797.000    3180.800
```

Figura 13

Os três testes seguintes avaliam as opções de conversão de unidades.

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -c l -b 10
NETIF                 TX          RX       TRATE       RRATE
lo                     0           0           0           0
wlp0s20f3       3716.000    4068.000     371.600     406.800
```

Figura 14

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -c l -k 10
NETIF                 TX          RX       TRATE       RRATE
lo                     0           0           0           0
wlp0s20f3         22.563       8.242       2.256        .824
```

Figura 15

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -c l -m 10
NETIF                 TX          RX        TRATE        RRATE
lo                     0           0            0            0
wlp0s20f3           .032        .010         .003         .001
```

Figura 16

De seguida temos a opção -p com argumento igual a 1, portanto só queremos que apareça uma interface na tabela.

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -c l -p 1 10
NETIF                 TX          RX        TRATE        RRATE
lo                     0           0            0            0
```

Figura 17

Para as opções de ordenação vamos obter respetivamente, sort (decrescente) pela coluna do TX, do TRATE, do RX e do RRATE.

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -c l -t 10
NETIF                 TX          RX        TRATE        RRATE
lo              4768.000    4768.000     476.800      476.800
wlp0s20f3      11538.000   22121.000    1153.800     2212.100
```

Figura 18

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -T 10
NETIF                 TX          RX        TRATE        RRATE
wlp0s20f3      89687.000   161372.000    8968.700    16137.200
lo               576.000      576.000      57.600       57.600
virbr0                 0            0            0            0
virbr0-nic             0            0            0            0
```

Figura 19

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -r 10
NETIF                 TX          RX        TRATE        RRATE
wlp0s20f3      59159.000    87243.000    5915.900     8724.300
lo                     0            0            0            0
virbr0                 0            0            0            0
virbr0-nic             0            0            0            0
```

Figura 20

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -R 10
NETIF                 TX          RX        TRATE        RRATE
wlp0s20f3      50811.000   73765.000     5081.100     7376.500
lo                     0           0            0            0
virbr0                 0           0            0            0
virbr0-nic             0           0            0            0
```

Figura 21

Seguidamente, verificamos o funcionamento da opção *loop*, que tal como esperado imprime a tabela de 10 em 10 segundos, atualizando os valores medidos.

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -l 10
NETIF                 TX          RX        TRATE        RRATE        TXTOT        RXTOT
lo                     0           0            0            0            0            0
virbr0                 0           0            0            0            0            0
virbr0-nic             0           0            0            0            0            0
wlp0s20f3      51545.000   108518.000     5154.500    10851.800     51545.000   108518.000

NETIF                 TX          RX        TRATE        RRATE        TXTOT        RXTOT
lo                410.000     410.000       41.000       41.000       410.000      410.000
virbr0                 0           0            0            0            0            0
virbr0-nic             0           0            0            0            0            0
wlp0s20f3      148858.000   104130.000    14885.800    10413.000    200403.000   212648.000

NETIF                 TX          RX        TRATE        RRATE        TXTOT        RXTOT
lo                     0           0            0            0      410.000      410.000
virbr0                 0           0            0            0            0            0
virbr0-nic             0           0            0            0            0            0
wlp0s20f3       96912.000   190395.000     9691.200    19039.500    297315.000   403043.000

NETIF                 TX          RX        TRATE        RRATE        TXTOT        RXTOT
lo                425.000     425.000       42.500       42.500       835.000      835.000
virbr0                 0           0            0            0            0            0
virbr0-nic             0           0            0            0            0            0
wlp0s20f3      138720.000   548650.000    13872.000    54865.000    436035.000   951693.000

NETIF                 TX          RX        TRATE        RRATE        TXTOT        RXTOT
lo                     0           0            0            0      835.000      835.000
virbr0                 0           0            0            0            0            0
virbr0-nic             0           0            0            0            0            0
wlp0s20f3       9770.000   117790.000      977.000    11779.000    445805.000  1069483.000

NETIF                 TX          RX        TRATE        RRATE        TXTOT        RXTOT
lo                216.000     216.000       21.600       21.600      1051.000     1051.000
virbr0                 0           0            0            0            0            0
virbr0-nic             0           0            0            0            0            0
wlp0s20f3       22326.000    67671.000     2232.600     6767.100    468131.000  1137154.000
```

Figura 22

Testámos também a combinação entre as opções loop, ordenação pelo TX e unidades em kilobytes, obtendo o resultado esperado.

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -l -t -k 10
NETIF                TX          RX          TRATE       RRATE       TXTOT       RXTOT
wlp0s20f3       183.856      40.028      18.385       4.002     183.856      40.028
virbr0-nic            0           0           0           0           0           0
virbr0                0           0           0           0           0           0
lo                    0           0           0           0           0           0

NETIF                TX          RX          TRATE       RRATE       TXTOT       RXTOT
wlp0s20f3        62.075      21.217       6.207       2.121     245.931      61.246
lo                 .181        .181        .018        .018        .181        .181
virbr0-nic            0           0           0           0           0           0
virbr0                0           0           0           0           0           0

NETIF                TX          RX          TRATE       RRATE       TXTOT       RXTOT
wlp0s20f3        57.045      82.483       5.704       8.248     302.977     143.729
lo                    0           0           0           0        .181        .181
virbr0-nic            0           0           0           0           0           0
virbr0                0           0           0           0           0           0

NETIF                TX          RX          TRATE       RRATE       TXTOT       RXTOT
wlp0s20f3         5.563      24.066        .556       2.406     308.541     167.795
lo                    0           0           0           0        .181        .181
virbr0-nic            0           0           0           0           0           0
virbr0                0           0           0           0           0           0

NETIF                TX          RX          TRATE       RRATE       TXTOT       RXTOT
wlp0s20f3         5.338      72.393        .533       7.239     313.879     240.189
lo                    0           0           0           0        .181        .181
virbr0-nic            0           0           0           0           0           0
virbr0                0           0           0           0           0           0

NETIF                TX          RX          TRATE       RRATE       TXTOT       RXTOT
wlp0s20f3         6.194      57.941        .619       5.794     320.074     298.130
lo                    0           0           0           0        .181        .181
virbr0-nic            0           0           0           0           0           0
virbr0                0           0           0           0           0           0
```

Figura 23

Por fim, repetimos o teste anterior adicionando a opção -v e confirmamos também o bom funcionamento da mesma.

```
raquel@raquel-Spin-SP314-52:~/Desktop/LEI/SO/Projeto/SO$ ./netifstat.sh -l -t -k -v 10
NETIF                TX            RX         TRATE         RRATE         TXTOT         RXTOT
virbr0                0             0             0             0             0             0
virbr0-nic            0             0             0             0             0             0
lo                 .400          .400          .040          .040          .400          .400
wlp0s20f3        44.284        27.141         4.428         2.714        44.284        27.141

NETIF                TX            RX         TRATE         RRATE         TXTOT         RXTOT
virbr0                0             0             0             0             0             0
virbr0-nic            0             0             0             0             0             0
lo                 .638          .638          .063          .063         1.039         1.039
wlp0s20f3        70.975        73.632         7.097         7.363       115.259       100.774

NETIF                TX            RX         TRATE         RRATE         TXTOT         RXTOT
virbr0                0             0             0             0             0             0
virbr0-nic            0             0             0             0             0             0
lo                    0             0             0             0         1.039         1.039
wlp0s20f3        80.924        57.865         8.092         5.786       196.184       158.639

NETIF                TX            RX         TRATE         RRATE         TXTOT         RXTOT
virbr0                0             0             0             0             0             0
virbr0-nic            0             0             0             0             0             0
lo                    0             0             0             0         1.039         1.039
wlp0s20f3        71.724        21.393         7.172         2.139       267.909       180.033

NETIF                TX            RX         TRATE         RRATE         TXTOT         RXTOT
virbr0                0             0             0             0             0             0
virbr0-nic            0             0             0             0             0             0
lo                    0             0             0             0         1.039         1.039
wlp0s20f3       163.139        31.637        16.313         3.163       431.048       211.670

NETIF                TX            RX         TRATE         RRATE         TXTOT         RXTOT
virbr0                0             0             0             0             0             0
virbr0-nic            0             0             0             0             0             0
lo                    0             0             0             0         1.039         1.039
wlp0s20f3       116.626        74.002        11.662         7.400       547.675       285.673
```

Figura 24

5 Conclusão

O script desenvolvido vai de encontro ao proposto, permitindo analisar a quantidade de dados transmitidos e recebidos e as respetivas taxas de transferência, para além de ser possível utilizar diferentes filtros de visualização.

Com efeito, a realização deste trabalho contribuiu positivamente para o aprofundamento dos nossos conhecimentos em *bash*, através da pesquisa que exigiu, já que ficámos a conhecer novos comandos (p.ex. o *basic calculator*) e estruturas de dados, como os *arrays* associativos. Além disso, tivemos também mais contacto com a manipulação de ficheiros através da linha de comandos, aumentando a nossa experiência nesse campo.

Durante a execução do trabalho tivemos de lidar com algumas dificuldades, nomeadamente perceber como acederíamos aos valores desejados e qual a melhor maneira de os tratar, mas com recurso aos guiões práticos e alguns sites, nomeadamente fóruns de perguntas, conseguimos chegar ao resultado pretendido.

6 Bibliografia

Para a realização deste trabalho consultamos os slides teóricos e os guiões práticos disponibilizados pelo docente, assim como os seguintes sites (consultados entre os dias 20/11/21 e 05/12/21):

- https://stackoverflow.com/
- https://www.computerhope.com/unix/bash/getopts.htm
- https://www.cyberciti.biz/faq/bash-get-basename-of-filename-or-directory-name/
- https://riptutorial.com/bash/example/31704/sort-by-keys