Monitorização de interfaces de rede em bash

universidade de aveiro

2021/2022
L. Engenharia Informática

Rafael Remígio 102435
João Correia 104360

Professor Nuno Lau
Professor Guilherme Campos

Índice

Introdução 2

Main 3

Looping 9

Functions 10

Testes Realizados 13

Bibliografia 19

Introdução

Neste documento, explicaremos a resolução do 1º Trabalho Prático da disciplina de Sistemas Operativos quem tem como objetivo o desenvolvimento e teste de um Script Bash capaz da monitorização de interfaces de rede apresentando esta informação em para o utilizador no terminal. Deve permitir a visualização de todas as interfaces de Rede disponíveis e o tamanho de pacotes transmitidos e recebidos durante um período de tempo imposto pelo Utilizador e respetivas taxas de transferências tal como ferramentas de Sorting e Seleção.

Explicaremos os métodos usados para cada função, tratamento de dados e ferramenta necessária para o funcionamento do programa seguindo a ordem de construção, tal como também mostraremos como foi o código foi testado e analisado de forma a eliminar erros.

Main

A partir do comando bash “ifconfig -a” recolhemos a informação necessária e organizamos esta data. Criamos 3 arrays intermédios onde iremos salvar os valores de RX, TX, e o nome das interfaces, usando awk grep e tr para a selecionar a informação necessária.

```
# get inicial array and organize it in: i_data[] = interface_name rx tx
IFS=$'\n' read -r -d '' -a interfaces < <( ifconfig -a | grep ": " | awk '{print $1}' | tr -d :  )
IFS=$'\n' read -r -d '' -a RXs < <( ifconfig -a | grep "RX packets" | awk '{print $5}' )
IFS=$'\n' read -r -d '' -a TXs < <( ifconfig -a | grep "TX packets" | awk '{print $5}' )
```

Organizamos em seguida um array “i_data” com três colunas divididas por um espaço (como uma matriz n por 3) capaz de ser percorrida a partir de coluna e linha através do awk e guardada toda a informação num só array, tal como ajuda na organização, Sorting, e no corte necessário de alguma interface.
Exemplo de como ficará organizada a informação:

```
i_data[$i]="${interfaces[$i]} ${RXs[$i]} ${TXs[$i]}";
```

```
eth0 0 0
lo 0 0
wifi0 0 0
```

Usamos de seguida o comando “sleep” com o último argumento imposto pelo Utilizador.

Mais uma vez a partir do comando “ifconfig -a” recolhemos a informação e organizando-a da mesma forma anterior neste caso guardando a no array “data”. Depois de termos os dois arrays fazemos a subtração dos valores dos RX e TX e salvamos na variável (“data”), tendo deste modo toda a informação necessária para escrever e fazer quaisquer cálculos ou sortings intermédios.

Argumentos passados pelo Utilizador

Iteramos os argumentos passados pelo Utilizador utilizando a instrução swicth (case), identificando os argumentos passados e dar erro se for introduzido um valor não especificado tal como verifica se o último argumento passado pelo utilizador é um número inteiro.

-c: muda o valor da variável booleana de modo que na próxima iteração leia o Regex introduzido pelo Utilizador

-p: muda o valor da variável booleana de modo que na próxima iteração leia o número de interfaces a mostrar só depois de efetuado o sorting para que as interfaces sejam escolhidas por ordem alfabética(default) ou pela ordem escolhida pelo Utilizador

-b/k/m: são opções que mudam o “byte_divisor”, uma variável que usamos para dividir os valores de RX e TX quando escrevemos a informação.

```
tx=$(echo "${data[$i]}" | awk '{print $2;}');
t_rate=$(bc <<<"scale=1;($tx/$sleep_time)/$byte_div");
tx=$((tx/byte_div));

rx=$(echo "${data[$i]}" | awk '{print $3;}');
r_rate=$(bc <<<"scale=1;($rx/$sleep_time)/$byte_div");
rx=$((rx/byte_div));
```

-r / t: Seleciona o uso da função SortRX e da função SortTX respetivamente através de um variável booleana. (Funções explicadas na secção sobre Funções)

-R / T: Seleciona o uso da função SortRX e da função SortTX respetivamente através de um variável booleana. (Funções explicadas na secção sobre Funções). Neste caso são selecionadas as mesmas funções que na seleção de r ou t pois o T_Rate e X_Rate irão sempre ser proporcionais aos valores de RX e TX

-v: Seleciona o uso da função Reverse através de um variável booleana. (Funções explicadas na secção sobre Funções)

```
# Sort
case $order in
    r)
        SortRx
        ;;
    t)
        SortTx
        ;;
esac
# Reverse
if [[ $reversed -eq 1 ]]
then
    Reverse
fi
# -p
if [ $N -gt $max ]; then
    N=$(($max));
fi
# -c
if [ ! -z "$Regex" ]; then
    #Regex will remove the ones that dont match regex pattern
    Regex
fi
```

-l: Muda o valor da variável booleana looping (explicado abaixo com mais detalhe)

Argumentos não reconhecidos e erros

Caso sejam passados argumentos não suportados, é apresentada a mensagem:

```
error
Argument exemplo is invalid
```

Caso o último argumento passado não seja um número inteiro, é apresentada a mensagem:

```
~./NetStat.sh -r
error
Last argument should be integer
```

Selecionando as opções -c e -p e o argumento seguinte será lido e introduzido numa variável $Regex e $max respetivamente. Se o valor introduzido a seguir a -p não for um inteiro é apresentada a seguinte mensagem de erro:

```
 ~./NetStat.sh -p palavra 2
error
Argument after -p should be integer
```

Escrever a tabela com os valores desejados

Cabeçalho:

Dependendo do valor da variável booleana “$looping” o cabeçalho poderá ser formatado das seguintes formas:

NETIF TX RX TRATE RRATE
ou
NETIF TX RX TRATE RRATE TXTOT RXTOT

```
if [ $loop -eq 1 ];
then
    printf "%-11s %9s %9s %9s %9s %9s %9s\n" "NETIF" "TX" "RX" "TRATE" "RRATE" "TXTOT" "RXTOT";
else
    printf "%-11s %9s %9s %9s %9s\n" "NETIF" "TX" "RX" "TRATE" "RRATE";
fi
```

Escrever os valores

Iteramos o array “data” recolhendo os valores de TX e RX de cada interface e inserindo-os num novo array “tx” e “rx”, TX_Rate e RX_Rate respetivamente, dividindo os valores inseridos pela variável “byte_division” explicada anteriormente e no caso de TX_Rate e RX_Rate dividimos também pelo nosso tempo de espera

```
int=$(echo "${data[$i]}" | awk '{print $1;}');

tx=$(echo "${data[$i]}" | awk '{print $2;}');
t_rate=$(bc <<<"scale=1;($tx/$sleep_time)/$byte_div");
tx=$((tx/byte_div));

rx=$(echo "${data[$i]}" | awk '{print $3;}');
r_rate=$(bc <<<"scale=1;($rx/$sleep_time)/$byte_div");
rx=$((rx/byte_div));
```

Depois de termos os estes arrays usamos o comando “printf” do bash imprimimos a informação com a formatação desejada.
Se for escolhida neste caso a opção -l serão também adicionadas as variáveis “$tmp_tot_tx” e “$tmp_tot_rx” (o seu uso será explicado na secção Looping)

```
    printf "%-11s %9s %9s %9s %9s %9s %9s\n" $int $tx $rx $t_rate $r_rate $tmp_tot_tx $tmp_tot_rx;
else
    printf "%-11s %9s %9s %9s %9s\n" $int $tx $rx $t_rate $r_rate;
```

As interfaces serão sempre escritas pela ordem existente no array “data”, logo todas as funções de ordenação e remoção e adição de interfaces é feita antes da escrita no terminal, através de funções como SortRX, SortTX, Reverse e Regex, e o número de interfaces a mostrar é definido pela variável “max”, inicializada na passagem de argumentos.

A ordem pela qual as interfaces são escritas é por predefinição a ordem alfabética, podendo ser mudada pelas opções -r, -t, -R e -T .

Looping

Caso a seja selecionada a função de loop (-l) o script será iterado sem terminar fim sendo só possível interrompe-lo forçando a paragem. Será visualizado pelo Utilizador a informação normal a visualizar tal como duas colunas com a informação do tamanho total de informação transmitida por interface. O script escrevera no ecrã em intervalos de tempo iguais ao introduzido pelo utilizador, incrementado a cada iteração o valor total das variáveis “tot_tx” e “tot_rx” e de seguida inserindo esta informação no array “data”.

```
if [ $iter -eq 0 ];
then
    tot_tx[$i]=$t_Gap;
    tot_rx[$i]=$r_Gap;

else

    tot_tx[$i]=$((tot_tx[$i]+$t_Gap));
    tot_rx[$i]=$((tot_rx[$i]+$r_Gap));
fi
# return changed and added values to the array
data[$i]="$interface $t_Gap $r_Gap ${tot_tx[$i]} ${tot_rx[$i]}";
```

A informação será apresentada através do uso de variáveis intermédias “tmp_tot_tx” e “tmp_tot_rx”

```
if [ $loop -eq 1 ];
then
    tmp_tot_tx=$(echo "${data[$i]}" | awk '{print $4;}');
    tmp_tot_tx=$((tmp_tot_tx/byte_div));
    tmp_tot_rx=$(echo "${data[$i]}" | awk '{print $5;}');
    tmp_tot_rx=$((tmp_tot_rx/byte_div));

    printf "%-11s %9s %9s %9s %9s %9s %9s\n" $int $tx $rx $t_rate $r_rate $tmp_tot_tx $tmp_tot_rx;
```

As funções Reverse, SortRX e SortTX são executadas em todas as iterações do código isto devido as mudanças no tamanho da informação recebida e transmitida por cada interface podendo ser ordenadas de formas diferentes dependendo dos valores recebidos.

Funções

SortTX() e SortRX()

As funções SortTX e SortRX ordenam o array “data” através de algoritmo de seleção das variáveis da tx ou rx respetivamente. O algoritmo de seleção ordena comparando cada elemento do array com os elementos seguintes posicionando o valor mais alto no index acima, correndo até terminar o array.

```
SortRx() {
    # basic selection sort
    for (( i = 0; $i < $(($N-1)); $((i = $i + 1)) ))
    do
        for (( e = $(($i+1)); $e < $N; $((e = $e + 1)) ))
        do

            rx_i=$(echo "${data[$i]}" | awk '{print $3;}');
            rx_e=$(echo "${data[$e]}" | awk '{print $3;}');

            if [[ $rx_e -gt $rx_i ]]
            then

                temp=${data[$e]};
                data[$e]=${data[$i]};
                data[$i]=$temp;

            fi
        done
    done
}
```

Reverse()

A função Reverse inverte a ordem das interfaces no array “data” trocando os indexes das entradas no array com a mesma distância ao centro.

```
Reverse(){
    # reverse order
    for (( i = 0; $i <= $(($N - $i - 1)); $((i = $i + 1)) ))
    do
        f=$(($N - $i - 1));

        temp=${data[$f]};
        data[$f]=${data[$i]};
        data[$i]=$temp;

    done
}
```

Regex()

A função Regex itera o array “data” recolhendo o nome da interface e comparando-a à expressão regular introduzida pela Utilizador nos argumentos da função. Se o nome da interface corresponder à expressão regular então a interface será adiciona a um array intermédio que depois será passado como array “data” original. Precisamos, depois de retirar as interfaces que não correspondem ao “regex”, mudar o número de iterações a correr quando escrevemos no ecrã.

```
Regex(){
    #Runs trough all of the interfaces and checks wich ones contain the regex in question
    for (( i=0; i < $N; i++ ))
    do
        interface_name=$(echo "${data[$i]}" | awk '{print $1;}');
        if [[ ${interface_name} =~ ^$Regex$ ]]; then
            new_array+=( "${data[i]}" )                 # adds the interface in a intermidiate array
        fi
    done

    data=("${new_array[@]}")        # reset the array data
    unset new_array                 #unsets the array since it has no more use

    #reset the number of interfaces
    N=${#data[@]};

}
```

Conclusão

A função **netifstat.sh** foi implementada com sucesso, mostrando todos os resultados pretendidos e implementadas todas as opções e funcionalidades pedidas pelo enunciado. Este trabalho levou-nos a aprender e implementar algumas ferramentas do **bash,** tal como o **awk, grep, sleep** e **printf.** Maior parte do conhecimento necessário para a completação do script foi adquirido nas aulas práticas através de guiões anteriores ou duvidas retiradas ao professor presente, tal como através do uso de fórums ou de documentação online e claro o manual do **bash.**

Estamos satisfeitos com a nossa implementação do programa apesar de nos apercebermos que ainda existe muitas formas de melhorar o uso de memoria tal existem algumas redundâncias e implementações pouco eficientes.

Tendo realizado os testes e tendo estes dar os resultados esperados e bem-sucedidos damos o trabalho realizado com êxito.

Ambos tivemos o mesmo nível de envolvimento e aprendizagem no trabalho. Deste modo a nota deverá ser repartida de igual forma.

Teste realizados

```
$./NetStat.sh 10
NETIF          TX          RX      TRATE      RRATE
bond0           0           0          0          0
dummy0          0           0          0          0
eth0          980        1066       98.0      106.6
lo              0           0          0          0
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 01.

```
$./NetStat.sh -c ".*0" 10
NETIF          TX          RX      TRATE      RRATE
bond0           0           0          0          0
dummy0          0           0          0          0
eth0         1022        1108      102.2      110.8
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 02.

```
$./NetStat.sh -p 3 10
NETIF          TX          RX      TRATE      RRATE
bond0           0           0          0          0
dummy0          0           0          0          0
eth0          980        1840       98.0      184.0
```

Figura 03

```
$./NetStat.sh -p 3 -r 10
NETIF          TX          RX      TRATE      RRATE
eth0            0         176          0       17.6
dummy0          0           0          0          0
bond0           0           0          0          0
```

Figura 04

```
$./NetStat.sh -b 10
NETIF          TX          RX      TRATE      RRATE
bond0           0           0          0          0
dummy0          0           0          0          0
eth0         1022        1022      102.2      102.2
lo              0           0          0          0
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 05

```
$./NetStat.sh -k 10
NETIF          TX          RX      TRATE      RRATE
bond0           0           0          0          0
dummy0          0           0          0          0
eth0            0           1          0         .1
lo              0           0          0          0
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 06

```
$./NetStat.sh -m 10
NETIF          TX          RX      TRATE      RRATE
bond0           0           0          0          0
dummy0          0           0          0          0
eth0            0           0          0          0
lo              0           0          0          0
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 07

```
$./NetStat.sh -r 10
NETIF          TX          RX      TRATE      RRATE
eth0         1022        1108      102.2      110.8
dummy0          0           0          0          0
bond0           0           0          0          0
lo              0           0          0          0
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 08

```
$./NetStat.sh -R 10
NETIF          TX          RX      TRATE      RRATE
eth0          980        2644       98.0      264.4
dummy0          0           0          0          0
bond0           0           0          0          0
lo              0           0          0          0
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 09

```
$./NetStat.sh -t 10
NETIF          TX          RX      TRATE      RRATE
eth0          980        1555       98.0      155.5
dummy0          0           0          0          0
bond0           0           0          0          0
lo              0           0          0          0
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 10

```
$./NetStat.sh -T 10
NETIF          TX          RX      TRATE      RRATE
eth0          980        1400       98.0      140.0
dummy0          0           0          0          0
bond0           0           0          0          0
lo              0           0          0          0
sit0            0           0          0          0
tunl0           0           0          0          0
```

Figura 11

```
$./NetStat.sh -v 10
NETIF          TX          RX       TRATE       RRATE
tunl0           0           0           0           0
sit0            0           0           0           0
lo              0           0           0           0
eth0          980        1486        98.0       148.6
dummy0          0           0           0           0
bond0           0           0           0           0
```

Figura 12

```
$./NetStat.sh -l 10
NETIF          TX          RX       TRATE       RRATE      TXTOT      RXTOT
bond0           0           0           0           0          0          0
dummy0          0           0           0           0          0          0
eth0         1022        1862       102.2       186.2       1022       1862
lo              0           0           0           0          0          0
sit0            0           0           0           0          0          0
tunl0           0           0           0           0          0          0

bond0           0           0           0           0          0          0
dummy0          0           0           0           0          0          0
eth0          980        1066        98.0       106.6       2002       2928
lo              0           0           0           0          0          0
sit0            0           0           0           0          0          0
tunl0           0           0           0           0          0          0

bond0           0           0           0           0          0          0
dummy0          0           0           0           0          0          0
eth0          980        2642        98.0       264.2       2982       5570
lo              0           0           0           0          0          0
sit0            0           0           0           0          0          0
tunl0           0           0           0           0          0          0

bond0           0           0           0           0          0          0
dummy0          0           0           0           0          0          0
eth0          980        1195        98.0       119.5       3962       6765
lo              0           0           0           0          0          0
sit0            0           0           0           0          0          0
tunl0           0           0           0           0          0          0
```

Figura 13

```
$./NetStat.sh -l -r -p 3 -b -c ".*0" 10
NETIF          TX          RX      TRATE      RRATE      TXTOT      RXTOT
eth0         1022        1022      102.2      102.2       1022       1022
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0          980         980       98.0       98.0       2002       2002
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0          980        1341       98.0      134.1       2982       3343
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0         1022        1022      102.2      102.2       4004       4365
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0          980        2212       98.0      221.2       4984       6577
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0         1022        1528      102.2      152.8       6006       8105
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0
```

Figura 14

```
$./NetStat.sh -l -r -p 3 -k 15
NETIF          TX          RX      TRATE      RRATE      TXTOT      RXTOT
eth0            1           6          0         .4          1          6
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0            1           2         .1         .1          2          8
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0            1           2         .1         .1          4         11
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0            1           1          0          0          5         13
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0            1           3          0         .2          7         16
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0

eth0            1           1         .1         .1          8         18
dummy0          0           0          0          0          0          0
bond0           0           0          0          0          0          0
```

Figura 15

```
[sop0308@l040101-ws08 Desktop]$ ./NetStat.sh -p 3 2
NETIF         TX         RX      TRATE      RRATE
enp0s25          0        849          0      424.5
lo               0          0          0          0
virbr0           0          0          0          0

[sop0308@l040101-ws08 Desktop]$ ./NetStat.sh -p 3 -c ".*0.*" 2
NETIF         TX         RX      TRATE      RRATE
enp0s25          0        555          0      277.5
virbr0           0          0          0          0
```

Figura 16

```
[sop0308@l040101-ws08 Desktop]$ ./NetStat.sh -p 3 -c ".*0.*" -l  2
NETIF         TX         RX      TRATE      RRATE      TXTOT      RXTOT
enp0s25      204        900      102.0      450.0        204        900
virbr0         0          0          0          0          0          0

enp0s25      204       1354      102.0      677.0        408       2254
virbr0         0          0          0          0          0          0

enp0s25      204        332      102.0      166.0        612       2586
virbr0         0          0          0          0          0          0

enp0s25      204        651      102.0      325.5        816       3237
virbr0         0          0          0          0          0          0

enp0s25      204        422      102.0      211.0       1020       3659
virbr0         0          0          0          0          0          0

enp0s25      204        479      102.0      239.5       1224       4138
virbr0         0          0          0          0          0          0

enp0s25      204        588      102.0      294.0       1428       4726
virbr0         0          0          0          0          0          0
```

Figura 17

```
[sop0308@l040101-ws08 Desktop]$ ./NetStat.sh -p 3 -c ".*0.*" -l -k 2
NETIF         TX         RX      TRATE      RRATE      TXTOT      RXTOT
enp0s25          0          0         .1         .2          0          0
virbr0           0          0          0          0          0          0

enp0s25          0          0         .1         .1          0          0
virbr0           0          0          0          0          0          0

enp0s25          0          0         .1         .1          0          1
virbr0           0          0          0          0          0          0

enp0s25          0          0         .1         .3          0          1
virbr0           0          0          0          0          0          0

enp0s25          0          0         .1         .1          1          2
virbr0           0          0          0          0          0          0

enp0s25          0          1         .1         .5          1          3
virbr0           0          0          0          0          0          0

enp0s25          0          0         .1         .1          1          3
virbr0           0          0          0          0          0          0

enp0s25          0          0         .1         .1          1          3
virbr0           0          0          0          0          0          0

enp0s25          0          1         .1         .5          1          4
virbr0           0          0          0          0          0          0
```

Figura 18

Mensagens de erro

```
$./NetStat.sh
error
Last argument should be integer
```

Figura 19

```
$./NetStat.sh -c ".*0"
error
Last argument should be integer
```

Figura 20

```
$./NetStat.sh -p -r 2
error
Argument after -p should be integer
```

Figura 21

```
$./NetStat.sh arg -r 2
error
Argument arg is invalid
```

Figura 22

Bibliografia

https://stackoverflow.com/

https://www.gnu.org/software/bash/manual/bash.html

https://www.gnu.org/software/gawk/manual/gawk.html

https://linuxconfig.org/

Trabalho Realizado por:

Rafael Remígio 102435
João Correia 104360

universidade de aveiro