Relatório do 1º projeto de Sistemas Operativos (SO)

Projeto: **Monitorização de interfaces de rede em bash.**

Grupo: Guilherme Dias (nº 103128)
Tomás Almeida (nº 103300)

Data de entrega: 6 de Dezembro 2021

Docentes: Nuno Lau(TP) e Guilherme Campos(P)

Índice

Introdução

Este trabalho teve como objetivo desenvolvimento de um script em `bash` que apresenta estatísticas e dados sobre a quantidade de dados transmitidos e recebidos em interfaces de rede selecionadas, e sobre as respectivas taxas de transferência.

O trabalho teve como base o comando “netstat -ie” que nos fornece informações sobre as interfaces de rede do computador, e as estatísticas dos dados recebidos e transmitidos por essas mesmas interfaces desde o início da sessão .

Explicação das opções

O script “netifstat” tem como parâmetro obrigatório o período de tempo em que iremos visualizar os dados.
Apenas inserindo o tempo na execução do script, visualizamos então os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em bytes).

Através de um getopts é então feito o reconhecimento das opções introduzidas ao executar o script.

Opção: -c “regex”

A opção -c requer um argumento obrigatório, sendo este uma expressão regular para filtrar as interfaces de rede a visualizar pelo nome. O programa retornará os nomes das interfaces que cumprem o padrão inserido, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em bytes).
Caso não existam interfaces que satisfaçam a expressão regular, o programa retorna uma mensagem a dizer que não foram encontradas interfaces segundo o padrão introduzido.
Caso, ao executar o script com a opção -c, não se indique o padrão, o programa retorna uma mensagem a dizer que não foi passado nenhum padrão regex.

Opção: -b “visualize in bytes”

A opção -b não tem argumentos, o programa retornará a mesma informação de quando não se seleciona nenhuma opção.

Opção: -k “visualize in kilobytes"

A opção -k não tem argumentos, o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em kilobytes).

Opção: -m “visualize in megabytes"

A opção -m não tem argumentos, o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em megabytes).

Opção: -p “select number of interfaces"

A opção -p tem como argumento obrigatório o número de interfaces que se deseja visualizar , o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates. Apenas se visualizará o número de interfaces que o utilizador deu input, ou seja, se o utilizador passar “-p 2”, apenas serão retornados dois interfaces (toda a informação em bytes).

Opção: -t “sort by TX"

A opção -t não tem argumentos, o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em bytes), apresentando as interfaces por ordem decrescente do TX, desde a que tem o TX maior até à que tem o TX menor.

Opção: -r “sort by RX"

A opção -r não tem argumentos, o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em bytes), apresentando as interfaces por ordem decrescente do RX, desde a que tem o RX maior até à que tem o RX menor.

Opção: -T “sort by TRATE"

A opção -T não tem argumentos, o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em bytes), apresentando as interfaces por ordem decrescente do TRATE, desde a que tem o TRATE maior até à que tem o TRATE menor.

Opção: -R “sort by RRATE"

A opção -R não tem argumentos, o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em bytes), apresentando as interfaces por ordem decrescente do RRATE, desde a que tem o RRATE maior até à que tem o RRATE menor.

Opção: -v “reverse order"

A opção -R não tem argumentos, o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), e os respectivos rates (toda a informação em bytes), apresentando as interfaces por ordem decrescente do RRATE, desde a que tem o RRATE maior até à que tem o RRATE menor.

Opção: -l “work in loop"

A opção -l não tem argumentos, o programa retornará os nomes das interfaces, os dados recebidos (RX) e transmitidos (TX), os respectivos rates, e o TXTOTAL e RXTOTAL (toda a informação em bytes).

De x em x tempo o script retornará a informação , somando sempre ao TXTOTAL e ao RXTOTAL os valores de TX e RX.

Explicação do código

No começo do script é definida a função “getRules”, que será utilizada quando não é inserida nenhuma opção válida, imprimindo assim todas as opções reconhecíveis.

```
getRules(){

    echo " Invalid Option
    -c    : regex
    -b    : visualize in bytes
    -k    : visualize in kylobytes
    -m    : visualize in megabytes
    -p    : select number of interfaces
    -t    : sort by TX
    -r    : sort by RX
    -T    : sort by TRATE
    -R    : sort by RRATE
    -v    : reverse order
    -l    : work in loop
        "
}
```

Depois da função, são declarados todos os arrays e variáveis que serão utilizados ao longo do script.

De seguida é feita a verificação da introdução do parâmetro “tempo” (time), e o registo na variável “NINTERFACES” o número de interfaces de rede disponíveis.
Se não se verificar a existência da variável tempo, será imprimido no terminal uma mensagem de erro e o programa terminará assim.

```
if ! [[ ${@: -1} =~ $re ]] ; then
    echo "error: No time input" >&2; exit 1
elif  [[ ${@: -1} =~ $re ]] && [[ ${@: -2} =~ $p ]] ; then
   echo "error: No time input" >&2; exit 1

else
    time=(${@: -1});
fi


NINTERFACES=$(netstat -i | awk '{print $NF}' | wc -w)-1
NINTERFACES=$(($NINTERFACES - 1)) # numero de interfaces no pc retornadas pelo comando netstat
```

Depois disto é recolhido o TX e o RX de cada interface antes e depois de um “sleep $time”. É calculado e colocado em arrays os nomes, o TX(TX final - TX inicial) e o RX(RX final - RX inicial), e o TRATE ((TX final - TX inicial)/time) e RRATE ((RX final - RX inicial)/time) de cada interface, em bytes.
Para obter a informação em kilobytes ou em megabytes foi usado o mesmo método, apenas se utilizou as variáveis “kb” e “mb” para passar os dados de bytes para kilobytes ou megabytes, respectivamente.

É dentro do “getopts” que é realizada a maioria do código. Dependendo da opção introduzida será realizado o cálculo das variáveis de modo diferente, como já foi explicado anteriormente.

Em cada opção é realizado o print da tabela formatada, e depois o print dos dados que se querem disponibilizar.

Nas opções de “sort” (-t, -r, -T, -R), usando como exemplo a opção -t, é criado um array onde se colocam as variáveis do array que contém os valores de TX(arrayTXFB), por ordem decrescente, e depois comparando com os valores do “array TXFB”, será realizado o print ou não.

```
if [[ $sorting = "s" ]];then
    echo "ERROR : More than 2 sorting options were selected"
    exit 1;
fi

declare -a arrayTempNames=()
arrayTXTemp=($(tr ' ' '\n' <<<"${arrayTXFB[@]}" | sort -nr))

fmt="%-12s%-12s%-12s%-12s%-12s%-12s\n"
printf "$fmt" NETIF TX RX TRATE RRATE
for a in "${!arrayTXTemp[@]}"; do
    for i in "${!arrayTXFB[@]}"; do
        if [[ "${arrayTXTemp[$a]}" = "${arrayTXFB[$i]}" ]];then
            if [[ ! " ${arrayTempNames[*]} " =~ " ${arrayName[$i]} " ]]; then
                printf "$fmt" "${arrayName[$i]}" "${arrayTXFB[$i]}" "${arrayRXFB[$i]}" "${arrayTRATEB[$i]}" "${arrayRRATEB[$i
                arrayTempNames+=${arrayName[$i]}
            fi
        fi
    done
done
```

Este processo é semelhante para todas as outras opções de “sort”.

Na opção de “loop” (-l) é calculada toda a informação em bytes sempre que o ciclo while é iterado, apenas adicionando o campo “TTOTAL” e “RTOTAL”, que é a soma dos TX e RX de cada interface ao seu respectivo total, sempre que é realizado um novo loop.

Dificuldade maior: Não conseguimos fazer com que fosse possível introduzir mais do que uma opção de forma funcional.

Testes

Os testes que realizamos ao script foram baseados na execução do programa várias vezes,correndo por vezes no background um speedtest do wifi (de modo a obter valores mais altos nos parâmetros.), alterando as opções introduzidas e tentando verificar se o output era o esperado.

Realizamos estes mesmos testes na sala das aulas práticas (4.1.01), encontrando alguns erros no código que não davam problemas nos nossos computadores, nomeadamente nos “sorts” e na opção de “loop”.

Temos aqui então exemplo dos testes:

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -b 10
NETIF       TX          RX          TRATE       RRATE
enp7s0      0           0           0           0
lo          2229        2229        222         222
wlp0s20f    15123       19827       1512        1982
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -v 10
NETIF       TX          RX          TRATE       RRATE
wlp0s20f    993         1371        99          137
lo          2085        2085        208         208
enp7s0      0           0           0           0
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -k 10
NETIF       TX          RX          TRATE       RRATE
enp7s0      0           0           0           0
lo          70          70          7           7
wlp0s20f    3032        656158      303         65615
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -m 10
NETIF       TX          RX          TRATE       RRATE
enp7s0      0           0           0           0
lo          0           0           0           0
wlp0s20f    3           622         0           62
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -c "wl.*" 10
NETIF       TX          RX          TRATE       RRATE
wlp0s20f    4093        2491        409         249
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -p 2 10
NETIF       TX          RX          TRATE       RRATE
enp7s0      0           0           0           0
lo          4077        4077        407         407
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -t 10
NETIF       TX          RX          TRATE       RRATE
wlp0s20f    81669       16667       8166        1666
lo          4077        4077        407         407
enp7s0      0           0           0           0
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -r 10
NETIF       TX          RX          TRATE       RRATE
wlp0s20f    118592      4248571     11859       424857
lo          5039        5039        503         503
enp7s0      0           0           0           0
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -T 10
522 241 0
NETIF       TX          RX          TRATE       RRATE
wlp0s20f    5229        1824        522         182
lo          2417        2417        241         241
enp7s0      0           0           0           0
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -R 10
NETIF       TX          RX          TRATE       RRATE
lo          2437        2437        243         243
wlp0s20f    1457        1871        145         187
enp7s0      0           0           0           0
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -R 10
NETIF       TX          RX          TRATE       RRATE
lo          2437        2437        243         243
wlp0s20f    1457        1871        145         187
enp7s0      0           0           0           0
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$
```

```
guidias@guidias-Legion-Y540-15IRH-PG0:~/GitHub/SOProject1$ ./netifstatFinal.sh -l 10
NETIF       TX          RX          TRATE       RRATE       TXTOT       RXTOT
enp7s0      0           0           0           0           0           0
lo          2085        2085        208         208         2085        2085
wlp0s20f    3028        4557        302         455         3028        4557


enp7s0      0           0           0           0           0           0
lo          3137        3137        313         313         5222        5222
wlp0s20f    29566       26091       2956        2609        32594       30648


enp7s0      0           0           0           0           0           0
lo          2085        2085        208         208         7307        7307
wlp0s20f    1402        2733        140         273         33996       33381


enp7s0      0           0           0           0           0           0
lo          2085        2085        208         208         9392        9392
wlp0s20f    12302       3180        1230        318         46298       36561


enp7s0      0           0           0           0           0           0
lo          5171        5171        517         517         14563       14563
wlp0s20f    98667       57537       9866        5753        144965      94098
```

Conclusão

A realização deste trabalho deu-nos a conhecer melhor o funcionamento do “awk” e do comando “netstat”.
Ficamos a perceber melhor o funcionamento da bash e todos os meandros de escrever um script em bash.
Reconhecemos que através de scripts em bash podemos obter variadas informações relevantes sobre o funcionamento do sistema operativo.

Bibliografia

https://www.cyberciti.biz/faq/bash-scripting-using-awk/

https://docs.oracle.com/cd/E19504-01/802-5753/6i9g71m3i/index.html

https://stackoverflow.com/questions/16483119/an-example-of-how-to-use-getopts-in-bash