Monitorização de redes em bash

Trabalho realizado por:

- Tiago Santos 95584
- Vasco Costa 97746

O problema proposto consistia na realização de um *script* em *Bourne-Again SHEll* (Bash) que permitisse realizar a monitorização das várias interfaces de rede permitindo a seleção de várias opções ao utilizador sendo estas escolhidas através da passagem de argumentos ao *script*, a unidade em que os valores são apresentados, ordem pela qual os mesmos são apresentados, se apenas uma única medição ou se em modo *loop* e os adaptadores que apresentam o nome que está em conformidade com uma expressão *regex*.

A solução encontrada consiste em primeiramente guardar os nomes dos adaptadores num *array* através do comando *ifconfig* sendo que é de seguida chamado outra vez para a leitura dos valores de *TX* e outra vez para os valores de *RX*, sendo os valores em *bytes* guardados cada um em seu *array*. Esta abordagem não é a mais desejada por se efetuarem demasiadas chamadas ao comando *ifconfig* e por se lerem em momentos diferentes os valores, no entanto, esta foi a única solução encontrada que funcionava no sistema operativo da sala 101, pois nos computadores pessoais com o *Ubuntu 20.04* conseguiu-se realizar o armazenamento do *output* do comando efetuando-se seguidamente a leitura e armazenamento em *arrays* da mesma forma.

Neste momento também são definidos os valores predefinidos para variáveis globais, que se tornam relevantes para quando não se pretendem utilizar as opções. Seguidamente chama-se a função de leitura dos valores atuais que inicialmente espera o tempo pretendido para intervalo entre medições. A leitura dos valores dos adaptadores é feita da mesma maneira que os primeiros valores. Efetuando-se as operações aritméticas de diferença entre os valores recebidos e transmitidos. Aqui já se faz uso de uma variável definida anteriormente, *is_loop* que será sempre 0 à primeira chamada desta função, servindo para se guardarem os primeiros valores medidos que serão úteis para se for pedido em *loop*.

Fez-se uso do comando *getopts* num ciclo para ler os diversos argumentos que foram passados ao *script*, tendo em consideração as opções que permitem a passagem de algum parâmetro, alternado o valor das variáveis que se encontravam com os valores predefinidos.

Caso não se tenha selecionado a opção de imprimir em *loop* imprimem-se os cabeçalhos da tabela sem as colunas da soma e seguidamente faz-se um ciclo para ler os valores de cada *array* e fazer a divisão por *num_to_divide* de forma a se conseguir ter na unidade pretendida, sem qualquer seleção estes irão aparecer em *bytes* mas se for pretendido outra unidade, o valor de *num_to_divide* é alterado de 1 para 1024 caso seja em *kilobytes* ou para 1024*1024 se se pretender em *megabytes*. Utilizando uma sequência de *pipes* consegue-se fazer a ordenação e o tratamento das restantes opções, expressão *regex* e número de adaptadores.

Antes de se imprimir no ecrã faz-se a ordenação segundo a forma que é pedida, sendo ordenado alfabeticamente por defeito, mas caso seja outra opção selecionada é alterado o valor de *sort_way*. O valor é alterado consoante a coluna em que os valores são apresentados, em primeira tem-se o nome dos adaptadores, na segunda o valor de *TX* e assim sucessivamente. A ordenação é naturalmente feita por ordem decrescente, isto por ser como

é apresentada a solução no guião apesar de ter sido um ponto de dúvida pois a seta vertical colocada ao lado do parâmetro a ordenar pode ser lida como sendo de forma ascendente. Como última etapa de ordenação verifica-se se é pretendido fazer *reverse*, esta parte inclui-se fora do ciclo *while* de forma a se poder colocar este como sendo o último parâmetro de *sort_way*. Verifica-se quais dos nomes se encontram em conformidade com a expressão *regex* que é passada como argumento seguida da opção "-c", a alteração é feita sem qualquer restrição na seleção da opção, alterando o valor predefinido de ".", representante de todos os valores. Relativamente ao número de interfaces apenas após as outras ordenações é feita para se conseguir conjugar com múltiplas opções, caso a opção seja alterada do valor de todos os nomes para outro, este é verificado se de facto é um número. É relevante este ser o último passo a ser feito pois caso se pretenda ordenar e escolher 3 se apresentem os 3 primeiros ou os 3 últimos.

Para a situação em que se pretende visualizar em *loop* faz-se o acréscimo das colunas da soma, para este efeito faz-se a subtração dos valores atuais de transmissão e de receção com os primeiros medidos, poderia também ter-se definido uma variável que iria acumulando os valores. Seguidamente chama-se a função de leitura dos valores atuais, nesta função é feita a pausa, e faz-se a impressão dos valores, fazendo este ciclo infinitamente.

Testes

Para o teste do *script* utilizou-se o utilizador Sop0503 na sala 101. Verificando-se que o programa não apresentava erros, no entanto, devido a limitações de tempo e de disponibilidade da sala apenas nos computadores pessoais foi possível registarem-se os valores medidos.

Na Figura 1 verifica-se o *output* do *script test_1.sh* que apenas apresenta 3 comandos onde se chama o comando *ifconfig* de seguida o *script netifstat.sh* com a opção de aguardar 5 segundos, e de seguida chama-se outra vez o comando *ifconfig*, para se alterarem os valores abriu-se manualmente o browser Mozilla Firefox e uma página. Deste teste consegue-se verificar que as contas estão a ser feitas corretamente.

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::e67d:6a8d:215e:91b6  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:87:73:f9  txqueuelen 1000  (Ethernet)
        RX packets 1866  bytes 1596669 (1.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1350  bytes 176415 (176.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 364  bytes 36590 (36.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 364  bytes 36590 (36.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

NETIF          TX       RX    TRATE    RRATE
enp0s3      75466   562931  15093,0 112586,0
lo          14076    14076   2815,0   2815,0
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::e67d:6a8d:215e:91b6  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:87:73:f9  txqueuelen 1000  (Ethernet)
        RX packets 2546  bytes 2159600 (2.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1822  bytes 251881 (251.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 486  bytes 50666 (50.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 486  bytes 50666 (50.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 1 – Execução do código apresentado em baixo com a abertura do browser Mozilla Firefox e de uma página

```
#!/bin/bash
ifconfig
./netifstat.sh 5
ifconfig
```

As opções de *sort* também foram verificadas e apresentavam-se como funcionais, assim como a seleção do *loop* é feita como pretendida. E os valores da soma correspondentes.

Na Figura 2 apresentamos outro teste, *test_2.sh,* que se utilizou para verificar que se estava a realizar a aritmética corretamente para o valor da soma. Para isso chama-se o

comando *ifconfig* seguido do *script* com as opções de fazer em *reverse*, em *loop* com intervalo de leitura de 5 segundos assim como a conversão para *kilobytes*. Fez-se recurso do comando *timeout* para se conseguir sair do *loop* infinito, após 11 segundos para dar tempo de se mostrar duas vezes. No entanto, os valores de *ifconfig*, como é evocado a seguir não irá ter o mesmo valor, consegue-se, no entanto, ter uma ideia genérica sobre o correto funcionamento. Mais se deve também aos erros de arredondamento.

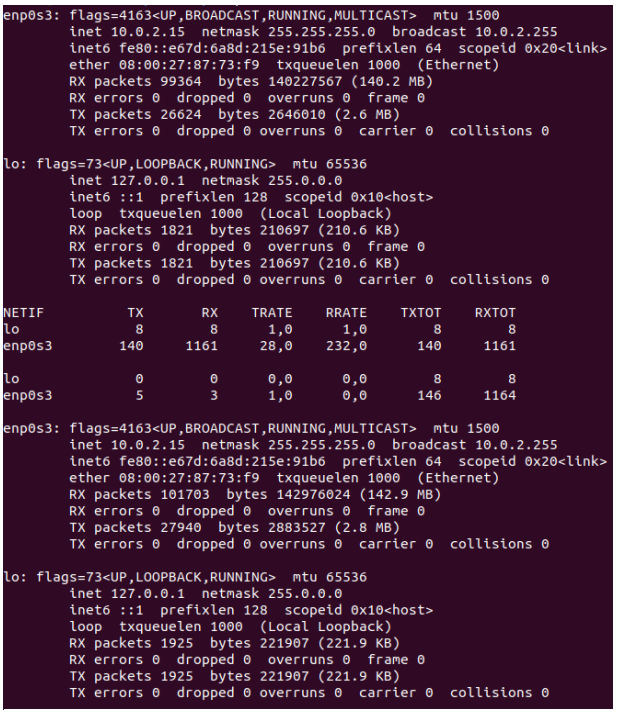

Figura 2– Execução do código apresentado em baixo com a visualização de um vídeo no YouTube

```
#!/bin/bash
ifconfig
timeout 11 ./netifstat.sh -l -k -v 5
ifconfig
```

Considerações finais

Consideramos que os resultados obtidos fazem sentido e que alcançámos com sucesso os objetivos. No entanto, as chamadas ao comando *ifconfig* deveriam ser duas, no entanto não conseguimos alcançar esse pretendido. Assim como poderia ser feito um tratamento de erros mais extenso, no entanto, esse é um processo itinerante em que é necessário pensar nas diversas formas que o utilizador poderia errar. No início deveria de ser feita a verificação da passagem de pelo menos um argumento sendo o último um número, no entanto, devido à impossibilidade de se testar esse funcionamento na sala 101 não se adicionou.

De forma a se simplificar a complexidade computacional do programa não foi permitida a ordenação múltipla, não se permite ordenar primeiramente por um parâmetro e conseguinte por outro, no entanto, esta opção era muito simplesmente implementada, seria apenas alterar `sort_way="-k2 "` para `sort_way="$sort_way""-k2 "` necessitando de se usar o parâmetro `"-s "` para se preservar a ordem em situação de valor igual. Permitindo assim a ordenação por outro parâmetro para quando existissem valores iguais, como este é também um caso que considerámos raro numa aplicação destas pois os valores teriam de ser iguais, sendo algo muito pouco provável não se considerou como sendo uma vantagem.