MONITORIZAÇÃO DE INTERFACES DE REDE EM BASH

2021/2022

Guilherme Costa Antunes
103600

Sistemas Operativos
Prof. Nuno Lau

deti universidade de aveiro

Índice

Introdução

Na sequência do trabalho proposto na unidade curricular de Sistemas Operativos, este relatório tem como objetivo descrever a abordagem utilizada na resolução da questão proposta, bem como foi feita a validação da solução final, que tem como objetivo mostrar diferentes dados estatísticos relativos ao envio e receção de dados por parte das interfaces de rede do computador selecionadas. Também serão demonstrados os tipos de erros que podem ocorrer mediante diferentes opções de entrada no programa.

Ao longo do documento estará descrita sucintamente a funcionalidade e o modo de execução de cada "bloco" de código utilizado na construção do programa pretendido bem como a maneira de como é feita a gestão de erros

Metodologia

- Declaração de Variáveis

```
6   vBi=()  #Array de armazenamento dos valores inicialmente capturados nas interfaces de rede
7   vBf=()  #Array de armazenamento dos valores capturados nas interfaces de rede após intervalo de tempo sT
8   vBr=()  #Array de armazenamento da diferença entre os valores inicial e depois de sT segundos
9   tots=() #Array de armazenamento de dados extra em situação de loop
10  pR=()   #Array de armazenamento dos nomes das interfaces de rede em análise
11  filter=()   #Array de armazenamento dos padrões inseridos através da opção -c
12  vlu=0   #Variável de controlo do número de opções de conversão
13  vld=0   #Variável de controlo do número de opções de odernamento
14  vlt=0   #Variável de controlo do número de opções de conversão
15  ctrl=0  #Variável de controlo de erros durante o getops
16  sT=1    #Variável de armazenamento do valor introduzido para sleep
17  p=-1    #Variável de armazenamento do número de interfaces a dar print
18  sort=0  #Variável de armazenamento da opção de ordenamento
19  d=1 #Variável de armazenamento do número de vezes que deve ser dividido o valor capturado para ser convertido
20  rev=0   #Variável de ativação do ordenamento reverso
21  loop=0  #Variável de ativação do loop
22  f=0 #Variável de ativação de aplicação dos filtros introduzidos através de -c
```

Figura 1 - Declaração de arrays e variáveis iniciais

Declaração de todas as variáveis e arrays necessários ao correto funcionamento do programa com os seus valores iniciais. Estas variáveis sofrerão alterações ao longo da execução do programa tendo em vista a impressão no formato pedido pelo utilizador. A função de cada variável está descrita em comentário na linha da mesma.

- Função inputs()

```
24  inputs(){
25      if ! [[ ${@: -1} =~ ^[0-9]+$ ]]; then
26          echo "ERRO: O último valor deve ser o argumento de sleep"
27          ctrl=1
28          exit 1
29      fi
```

Figura 2 - Função inputs() 1/4

A função inputs é a primeira a ser invocada na linha de execução e tem como principal objetivo realizar praticamente toda a validação dos argumentos e opções passadas aquando a execução do programa em linha de comando. A primeira validação que realiza é através de um "if" verificar se o último valor digitado para executar, o valor ${@: -1} é um valor numérico inteiro. Este valor vai ser usado para definir o intervalo de tempo entre cada captura de dados nas interfaces de rede. Em qualquer avaliação realizada dentro da função inputs() caso se verifique que o argumento passado não é válido o programa terminará através do comando "exit" com valor "1" e a variável "ctrl" será alterada para 1 para posterior termino do programa com o mesmo valor na função main().

```
30      while getopts ":p:c:bkmtrTRlv" opt; do
31          case $opt in
32              b)
33                  if [[ $vlu -gt 0 ]]; then
34                      echo "ERRO: Só deve ser utilizada uma opção de conversão. Usar apenas um dos argumentos (-b -k -m)"
35                      ctrl=1
36                      exit 1
37                  fi
38                  vlu=$((vlu+1))
39              ;;
40              k)
41                  if [[ $vlu -gt 0 ]]; then
42                      echo "ERRO: Só deve ser utilizada uma opção de conversão. Usar apenas um dos argumentos (-b -k -m)"
43                      ctrl=1
44                      exit 1
45                  fi
46                  vlu=$((vlu+1))
47                  d=1024
48              ;;
49              m)
50                  if [[ $vlu -gt 0 ]]; then
51                      echo "ERRO: Só deve ser utilizada uma opção de conversão. Usar apenas um dos argumentos (-b -k -m)"
52                      ctrl=1
53                      exit 1
54                  fi
55                  vlu=$((vlu+1))
56                  d=1048576
57              ;;
```

Figura 3 - Função inputs() 2/4

Dentro da função inputs() está implementado um *while getops* que permite percorrer parâmetro a parâmetro os valores que foram introduzidos para serem executados no programa. O programa irá procurar por parâmetros do tipo indicado na string ":p:c:bkmtrTRlv" que indica que todas as opções devem ter algo a antecede-las e as -p e -c devem ter um valor a suceder-lhes. Neste caso avaliamos se a opção enviada corresponde a alguma opção de conversão de unidade, -b para bytes, -k para kilobytes e -m para megabytes. Caso exista mais que uma opção de conversão nos valores de execução o programa terminará já que a variável “vlu” estará acima do valor permitido, que para a correta execução do programa deve ser “0”. Mediante a opção é guardado em d o valor pelo qual se deve dividir o valor capturado para que este fique na unidade indicada nos parâmetros de execução do programa.

```
58              t)
59                  if [[ $vld -gt 0 ]]; then
60                      echo "ERRO: Só deve ser utilizada uma opção de ordenamento. Usar apenas um dos argumentos (-t -r -T -R)"
61                      ctrl=1
62                      exit 1
63                  fi
64                  vld=$((vld+1))
65                  sort=1
66              ;;
67              r)
68                  if [[ $vld -gt 0 ]]; then
69                      echo "ERRO: Só deve ser utilizada uma opção de ordenamento. Usar apenas um dos argumentos (-t -r -T -R)"
70                      ctrl=1
71                      exit 1
72                  fi
73                  vld=$((vld+1))
74                  sort=2
75              ;;
76              T)
77                  if [[ $vld -gt 0 ]]; then
78                      echo "ERRO: Só deve ser utilizada uma opção de ordenamento. Usar apenas um dos argumentos (-t -r -T -R)"
79                      ctrl=1
80                      exit 1
81                  fi
82                  ((vld++))
83                  sort=3
84              ;;
85              R)
86                  if [[ $vld -gt 0 ]]; then
87                      echo "ERRO: Só deve ser utilizada uma opção de ordenamento. Usar apenas um dos argumentos (-t -r -T -R)"
88                      ctrl=1
89                      exit 1
90                  fi
91                  ((vld++))
92                  sort=4
93              ;;
```

Figura 4 - Função inputs() 3/4

Neste bloco avaliamos se alguma das opções enviadas corresponde a uma opção de ordenamento. Assim como nas opções de conversão a variável "vld" não permite que sejam usadas mais do que 1 opção de ordenamento. Mediante a opção escolhida é guardada na variável "sort" o tipo de ordem que desejamos que o nosso programa execute o output, sempre por ordem crecente de valores, -t para ordenar pelos valores da coluna TX do ouput, -r para a coluna RX, -T para a coluna TRATE e -R para a RRATE.

```
94              p)
95                  if ! [[ "$OPTARG" =~ ^[0-9]+$ ]]; then
96                      echo "O argumento de -$opt deve ser um número inteiro não negativo"
97                      ctrl=1
98                      exit 1
99                  fi
100                 p=$OPTARG
101             ;;
102             l)
103                 loop=1
104             ;;
105             v)
106                 rev=1
107             ;;
108             c)
109                 if [ "$OPTARG" == "" ]; then
110                     echo "O argumento de -$opt deve ser uma string não vazia"
111                     ctrl=1
112                     exit 1
113                 fi
114                 filter=($OPTARG)
115                 f=1
116             ;;
117             *)
118                 echo "ERRO: Opção -$OPTARG não implementada"
119                 ctrl=1
120                 exit 1
121             ;;
122             \?)
123                 echo "ERRO: Formato ou número de argumentos inválido. Argumentos possíveis (-b -k -m -t -r -T -R -l -v -c -p)"
124                 ctrl=1
125                 exit 1
126             ;;
127         esac
128     done
129 }
```

Figura 5 - Função inputs() 4/4

Nestas linhas de código vamos avaliar se a opção a passar neste step do *while* corresponde a -p, -l, -v, -c ou a alguma opção não implementada ou simplesmente inválida.

No caso de -p o programa irá verificar se a seguir a esta opção se encontra um valor inteiro não negativo. Em caso afirmativo será guardada em “p” o número de interfaces que deve ser imprimido após a aplicação de filtros, em caso negativo o programa terminará com o respetivo erro.

Se for uma opção do tipo -l ou -v o programa irá guardar a ativação dessa opção respetivamente nas variáveis “loop” e “rev".

Na situação de uma opção -c o programa procura imediatamente a seguir à “-c” uma string não vazia. Caso não seja vazia será guardado em “filter” os valores de pesquisa por nome de interfaces, separados por espaço entre diferentes valores de pesquisa, e será ativada a opção de filtro por nome com a variável “f”. Caso seja vazia o programa terminará.

Caso a opção enviada não corresponda a nenhuma das já avaliadas pelo *getops* o programa terminará com o respetivo erro assim como caso se verifique outro tipo de situação não prevista pelo programa.

- Função calcBytes()

```
131  calcBytes(){
132      pR=($(ifconfig | grep -oP "\w+(?=: )"))
133      vBi=($(ifconfig | grep -oP "(?<=bytes )\w+"))
134      sleep $sT
135      vBf=($(ifconfig | grep -oP "(?<=bytes )\w+"))
136
137      for i in ${!vBi[@]}; do
138          vBr[$i]="$((${vBf[i]}-${vBi[i]}))"
139      done
140      if [[ $loop -eq 2 ]]; then
141          for i in ${!vBr[@]}; do
142              tots[$i]="$((${tots[i]}+${vBr[i]}))"
143          done
144      elif [[ $loop -eq 1 ]]; then
145          for i in ${!vBr[@]}; do
146              tots[$i]=0
147          done
148          loop=2
149      fi
150      if [[ $p -gt ${#pR[@]} ]] || [ $p -lt 0 ]; then
151          p=${#pR[@]}
152      fi
153  }
```

Figura 6 - Função calcBytes()

A função calcBytes() tem como principal função guardar as capturas realizadas às interfaces de rede ativas todas que existem no computador. Na array "pR" são armazenados os nomes das interfaces através de um *grep* com scope para todos os elementos do output de *ifconfig* que respeitem o regex "\w+(?=:)", isto é, os valores antes de qualquer sequência de ":" e espaço, sequência essa que em todo o output de *ifconfig* só se verifica após os nomes das interfaces. Na arrays "vBi" e "vBf" são armazenados os valores que respeitem o regex "(?<=bytes)\w+", isto é, todos os valores de RX e TX, que no output de *ifconfig* se encontram antes de qualquer sequência de "bytes" e espaço. Enquanto que a "vBi" é preenchida o mais cedo possível, a "vBf" só o é depois de passado o tempo de sleep inserido no ultimo parâmetro da chamada do programa e depois de guardados os dados em "vBi". Em "vBr" vai ser guardada a diferença entre "vBf" e "vBi", para o mesmo índice em ambas, percorrendo os índices de "vBi" com um for. Caso a opção de "loop" esteja ativa e seja a primeira vez que é invocada a função calcBytes() a array "tots" será incializada com zeros até ao índice correspondente ao último índice de "vBr". Caso já não seja a primeira vez o programa irá acrescentar ao valor adicionando a cada índice de "tots" os novos valores capturados, calculando assim a soma total desde o inicio do loop daquelas colunas do output.

No fim da função é avaliado o valor de "p", caso este seja superior ao número de interfaces ou inferior a 0 este é alterado para o exato número de interfaces ativas.

- Função printInfo()

```
155 printInfo(){
156     if [[ $loop -eq 2 ]]; then
157         for i in ${!pR[@]}; do
158             if [[ $f -eq 1 ]]; then
159                 for e in ${!filter[@]}; do
160                     if [[ ${pR[i]} =~ ${filter[e]} ]]; then
161                         printf "%s\t%10.0f\t%10.0f\t%10.1f\t%10.1f\t%10.0f\t%10.0f\n" "${pR[i]}" $(echo "scale=2; ${vBr[i*2+1]}/$d" | bc)
162                         $(echo "scale=2; ${vBr[i*2]}/$d" | bc ) $(echo "scale=2; ${vBr[i*2+1]}/($d*$sT)" | bc)
163                         $(echo "scale=2; ${vBr[i*2]}/($d*$sT)" | bc) $(echo "scale=2; ${tots[i*2+1]}/$d" | bc)
164                         $(echo "scale=2; ${tots[i*2]}/$d" | bc)
165                     fi
166                 done
167             else
168                 printf "%s\t%10.0f\t%10.0f\t%10.1f\t%10.1f\t%10.0f\t%10.0f\n" "${pR[i]}" $(echo "scale=2; ${vBr[i*2+1]}/$d" | bc)
169                 $(echo "scale=2; ${vBr[i*2]}/$d" | bc ) $(echo "scale=2; ${vBr[i*2+1]}/($d*$sT)" | bc) $(echo "scale=2; ${vBr[i*2]}/($d*$sT)" | bc)
170                 $(echo "scale=2; ${tots[i*2+1]}/$d" | bc) $(echo "scale=2; ${tots[i*2]}/$d" | bc)
171             fi
172         done
173     else
174         for i in ${!pR[@]}; do
175             if [[ $f -eq 1 ]]; then
176                 for e in ${!filter[@]}; do
177                     if [[ ${pR[i]} =~ .*${filter[e]}.* ]]; then
178                         printf "%s\t%10.0f\t%10.0f\t%10.1f\t%10.1f\n" "${pR[i]}" $(echo "scale=2; ${vBr[i*2+1]}/$d" | bc)
179                         $(echo "scale=2; ${vBr[i*2]}/$d" | bc ) $(echo "scale=2; ${vBr[i*2+1]}/($d*$sT)" | bc)
180                         $(echo "scale=2; ${vBr[i*2]}/($d*$sT)" | bc)
181                     fi
182                 done
183             else
184                 printf "%s\t%10.0f\t%10.0f\t%10.1f\t%10.1f\n" "${pR[i]}" $(echo "scale=2; ${vBr[i*2+1]}/$d" | bc)
185                 $(echo "scale=2; ${vBr[i*2]}/$d" | bc ) $(echo "scale=2; ${vBr[i*2+1]}/($d*$sT)" | bc) $(echo "scale=2; ${vBr[i*2]}/($d*$sT)" | bc)
186             fi
187         done
188     fi
189 }
```

Figura 7 - Função printInfo() (Modificada para efeitos de legibilidade)

A função printInfo() estabelece a maneira como o output será feito mediante as opções inseridas aquando a ordem de execução do programa. Caso a opção de loop esteja desligada, assim como a de filtragem por nome será impresso no formato visto nas linhas 184 e 185 da figura 7, sendo que imediatamente antes de ser impressa a informação é realizada uma divisão entre os valores guardados em "vBr" e os guardados na variável "d" através do comando *bc*. No caso do filtro por nome estar ativo só serão impressas as informações das interfaces referenciadas através de uma comparação de regex antes de ser dado print. No caso de a opção de loop estar ativa serão adicionadas duas colunas ao output com os totais desde o início do loop e o programa correrá infinitamente até ordem de paragem imprimindo de x em x tempo a tabela com os filtros escolhidos aplicados, sendo x o valor de "sT" que corresponde ao último parâmetro da lista de inputs.

- Função print()

```
print(){
    if [[ $rev -eq 0 ]]; then
        case $sort in
            0) printInfo | sort -k1 | head -n $p;;
            1) printInfo | sort -k2 -n | head -n $p;;
            2) printInfo | sort -k3 -n | head -n $p;;
            3) printInfo | sort -k4 -n | head -n $p;;
            4) printInfo | sort -k5 -n | head -n $p;;
        esac
    else
        case $sort in
            0) printInfo | sort -k1 -r | head -n $p;;
            1) printInfo | sort -k2 -n -r | head -n $p;;
            2) printInfo | sort -k3 -n -r | head -n $p;;
            3) printInfo | sort -k4 -n -r | head -n $p;;
            4) printInfo | sort -k5 -n -r | head -n $p;;
        esac
    fi
}
```

Figura 8 - Função print()

A função print() é a única que dá ordem de execução à função printInfo(). A função em primeiro lugar analisa se a opção de reverse está ativa pela variável “rev". Depois mediante o valor guardado em “sort” dá ordem de impressão à função printInfo() com as condicionantes de esse print estar organizado pelo comando *sort* mediante a coluna k de referência correspondente à opção em guardada em “sort” e limitada ao número de interfaces guardadas em “p” através do comando *head* após a filtragem toda. Caso a opção de reverse esteja ativa será exatamente o mesmo mas no comando *sort* também constará a ordem “-r” de reverter a ordem.

- Função execInput()

```
execInput(){

    if [[ $loop -eq 2 ]]; then
        while true; do
            calcBytes
            if [[ $vlt -lt 1 ]]; then
                printf "%s\t%10s\t%10s\t%10s\t%10s\t%10s\t%10s\n" "NETIF" "TX" "RX" "TRATE" "RRATE" "TXTOT" "RXTOT"
                ((vlt++))
            else
                printf "\n"
            fi
            print
        done
    else
        calcBytes
        printf "%s\t%10s\t%10s\t%10s\t%10s\n" "NETIF" "TX" "RX" "TRATE" "RRATE"
        print
    fi

}
```

Figura 9 - Função execInput()

A função execInput() é a primeira a executar um print e a que realiza a execução do programa mediante as opções mais relevantes para o seu correto funcionamento. Assim se a variável “loop” estiver ativa e no estado “2” realiza-se um *while* que atualiza os dados das capturas de interfaces chamando a

função calcBytes() e dá ordem de impressão do cabeçalho da tabela de loop, sendo que só permite que este seja impresso uma vez. Após isto chama a função print() anteriormente explicada.

No caso de a opção loop estar desativada o programa limitar-se há a atualizar os dados com calcBytes(), a dar print do cabeçalho de execução única e a dar ordem de avanço à função print().

- Função main()

```
224  main(){
225      inputs "$@"
226      if [[ $OPTIND -ne $# ]]; then
227          echo "ERRO: Formato ou número de argumentos inválido. Argumentos possíveis (-b -k -m -t -r -T -R -l -v -c -p)"
228          exit 1
229      fi
230      if [[ $ctrl -eq 1 ]]; then
231          exit 1
232      fi
233      sT="${@: -1}"
234      calcBytes
235      execInput
236  }
237
238  main "$@"
```

Figura 10 - Função main()

A função main é a que controla o programa e a outra que valida os argumentos de entrada para complementar a inputs(). Após executar a função inputs enviando todos os inputs colocados no sistema aquando a execução do programa a função main() verifica se o número de inputs é igual ao número de opções válidas registadas *pelo getops*, em caso afirmativo continua a executar o programa, em caso negativo o programa para com código de *exit* "1". Caso durante a execução da inputs() tenha havido uma alteração do valor da variável "ctrl" para "1" o programa também irá parar a execução. Sendo este o último momento em que o programa pode ser parado , sem ser forçado a isso, é atualizado o valor de "sT" para fazer o tempo de *sleep* pretendido, são chamadas as funções calcBytes() para atualizar os dados sobre as interfaces de rede e a função execInput() para dar continuidade à execução natural.

Testes de Execução

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh 2
NETIF            TX              RX           TRATE           RRATE
enp0s25           0             595             0.0           297.5
lo              343             343           171.5           171.5
virbr0            0               0             0.0             0.0
```

Figura 11 – Teste com 2 segundos

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -b 2
NETIF            TX              RX           TRATE           RRATE
enp0s25           0             320             0.0           160.0
lo              340             340           170.0           170.0
virbr0            0               0             0.0             0.0
```

Figura 12 – Teste com conversão para bytes

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -k 2
NETIF            TX              RX           TRATE           RRATE
enp0s25           0               0             0.0             0.1
lo                0               0             0.2             0.2
virbr0            0               0             0.0             0.0
```

Figura 13 -Teste com conversão para kilobytes

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -p 2 1
NETIF            TX              RX           TRATE           RRATE
enp0s25           0            1530             0.0          1530.0
lo                0               0             0.0             0.0
```

Figura 14 – Teste com filtro das 2 primeiras interfaces de rede da ordem inicial

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -p 1 -v 2
NETIF            TX              RX           TRATE           RRATE
virbr0            0               0             0.0             0.0
```

Figura 15 – Teste com filtro da primeira interfaces de rede da ordem inicial reversa

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -t 2
NETIF            TX              RX           TRATE           RRATE
enp0s25           0             192             0.0            96.0
lo                0               0             0.0             0.0
virbr0            0               0             0.0             0.0
```

Figura 16 – Teste com ordenamento pela coluna TX

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -R 2
NETIF           TX              RX           TRATE           RRATE
lo               0               0             0.0             0.0
virbr0           0               0             0.0             0.0
enp0s25          0             663             0.0           331.5
```

Figura 17 – Teste com ordenamento pela coluna RRATE

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -v 2
NETIF           TX              RX           TRATE           RRATE
virbr0           0               0             0.0             0.0
lo               0               0             0.0             0.0
enp0s25         90             383            45.0           191.5
```

Figura 18 – Teste com ordenamento reverso da ordem inicial

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -v -r 2
NETIF           TX              RX           TRATE           RRATE
enp0s25          0             256             0.0           128.0
virbr0           0               0             0.0             0.0
lo               0               0             0.0             0.0
```

Figura 19 – Teste com ordenamento pela coluna RX reversamente

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -v -k 2
NETIF           TX              RX           TRATE           RRATE
virbr0           0               0             0.0             0.0
lo               0               0             0.0             0.0
enp0s25          0               3             0.0             1.4
```

Figura 20 – Teste com conversão a kilobytes com ordenamento reverso

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -c "e.* l.*" 2
NETIF           TX              RX           TRATE           RRATE
enp0s25          0            1722             0.0           861.0
lo               0               0             0.0             0.0
```

Figura 21 – Teste com filtro de pesquisa pelos nomes de regex "e." e "l.*"*

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -c "e.* l.*" -v 2
NETIF           TX              RX           TRATE           RRATE
lo               0               0             0.0             0.0
enp0s25          0             339             0.0           169.5
```

Figura 22 - Teste com filtro de pesquisa pelos nomes de regex "e." e "l.*" ordenados reversamente*

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -l 3
NETIF          TX              RX           TRATE           RRATE          TXTOT           RXTOT
enp0s25         0            1913             0.0           637.7              0            1913
lo              0               0             0.0             0.0              0               0
virbr0          0               0             0.0             0.0              0               0

enp0s25         0             569             0.0           189.7              0            2482
lo              0               0             0.0             0.0              0               0
virbr0          0               0             0.0             0.0              0               0

enp0s25         0             737             0.0           245.7              0            3219
lo              0               0             0.0             0.0              0               0
virbr0          0               0             0.0             0.0              0               0
```

Figura 23 – Teste de loop com 3 segundos de intervalo

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -l -v 2
NETIF          TX              RX           TRATE           RRATE          TXTOT           RXTOT
virbr0          0               0             0.0             0.0              0               0
lo              0               0             0.0             0.0              0               0
enp0s25         0             608             0.0           304.0              0             608

virbr0          0               0             0.0             0.0              0               0
lo              0               0             0.0             0.0              0               0
enp0s25         0             565             0.0           282.5              0            1173

virbr0          0               0             0.0             0.0              0               0
lo              0               0             0.0             0.0              0               0
enp0s25         0            1133             0.0           566.5              0            2306
```

Figura 24 – Teste com loop com 2 segundos de intervalo e ordem reversa

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -l -v -k 2
NETIF          TX              RX           TRATE           RRATE          TXTOT           RXTOT
virbr0          0               0             0.0             0.0              0               0
lo              0               0             0.0             0.0              0               0
enp0s25         0               0             0.0             0.1              0               0

virbr0          0               0             0.0             0.0              0               0
lo              0               0             0.0             0.0              0               0
enp0s25         0               0             0.0             0.2              0               1

virbr0          0               0             0.0             0.0              0               0
lo              0               0             0.0             0.0              0               0
enp0s25         0               0             0.0             0.1              0               1
```

Figura 25 – Teste com conversão para kilobytes, com loop 2 segundos de intervalo e ordem reversa

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -l -v -k -T 2
NETIF          TX              RX           TRATE           RRATE           TXTOT           RXTOT
virbr0          0               0             0.0             0.0               0               0
lo              0               0             0.0             0.0               0               0
enp0s25         0               0             0.0             0.2               0               0

virbr0          0               0             0.0             0.0               0               0
lo              0               0             0.0             0.0               0               0
enp0s25         0               2             0.0             0.8               0               2

virbr0          0               0             0.0             0.0               0               0
lo              0               0             0.0             0.0               0               0
enp0s25         0               1             0.0             0.3               0               2
```

Figura 26 – Teste com conversão para kilobytes, com loop 2 segundos de intervalo e ordenado pela coluna TRATE reversamente

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -l -v -k -T -c "v.* e.*" 2
NETIF          TX              RX           TRATE           RRATE           TXTOT           RXTOT
virbr0          0               0             0.0             0.0               0               0
enp0s25         0               0             0.0             0.2               0               0

virbr0          0               0             0.0             0.0               0               0
enp0s25         0               2             0.0             1.0               0               2

virbr0          0               0             0.0             0.0               0               0
enp0s25         0               0             0.0             0.1               0               3
```

Figura 27 - Teste com conversão para kilobytes, com filtro de pesquisa pelos nomes de regex "v." e "e.*", com loop 2 segundos de intervalo e ordenado pela coluna TRATE reversamente*

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -l -v -k -T -c "v.* e.*" -p 1 5
NETIF          TX              RX           TRATE           RRATE           TXTOT           RXTOT
virbr0          0               0             0.0             0.0               0               0

virbr0          0               0             0.0             0.0               0               0

virbr0          0               0             0.0             0.0               0               0

enp0s25         0               1             0.1             0.2               1               7

virbr0          0               0             0.0             0.0               0               0
```

Figura 28 - Teste com conversão para kilobytes, com filtro da primeira interfaces de rede, com filtro de pesquisa pelos nomes de regex "v." e "e.*", com loop 5 segundos de intervalo e ordenado pela coluna TRATE reversamente*

De verificar nesta imagem que a interface "enp0s25" passou para o topo da tabela em estado reverso e por isso foi apenas ela impressa já que estávamos a limitar a impressão a apenas 1 interface por step de loop, ao contrário do que aconteceu nos outros steps.

```
[sop0209@l040101-ws01 ~]$ ./netifstat.sh -l -k -T -c "v.* e.*" -p 1 5
NETIF           TX              RX           TRATE           RRATE           TXTOT           RXTOT
enp0s25          0               2             0.0             0.4               0               2

enp0s25          0               0             0.0             0.1               0               2

enp0s25          0               2             0.0             0.4               0               4

virbr0           0               0             0.0             0.0               0               0

enp0s25          0               2             0.0             0.5               0               8
```

Figura 29 - Teste com conversão para kilobytes, com filtro da primeira interfaces de rede, com filtro de pesquisa pelos nomes de regex "v." e "e.*", com loop 5 segundos de intervalo e ordenado pela coluna TRATE*

De verificar nesta imagem que a interface "virbr0" passou para o topo da tabela e por isso foi a interface impressa já que estávamos a limitar a impressão a apenas 1 interface por step de loop, ao contrário do que aconteceu nos outros steps.

Erros

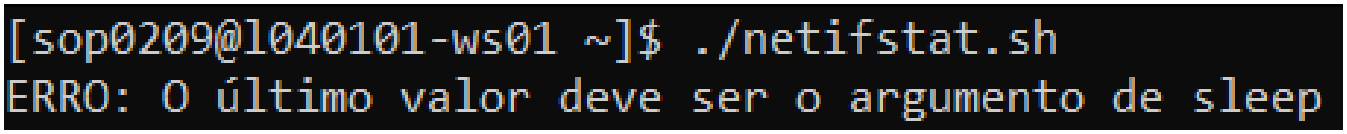

Figura 30 – Erro de não inserção de valor inteiro no último argumento de execução

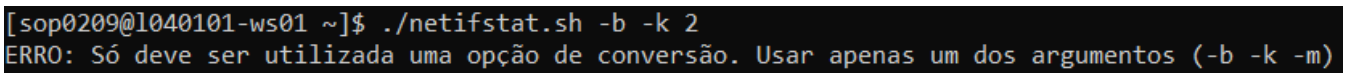

Figura 31 – Erro de inserção de mais do que um argumento de conversão

Figura 32 - Erro de inserção de mais do que uma opção de ordenamento

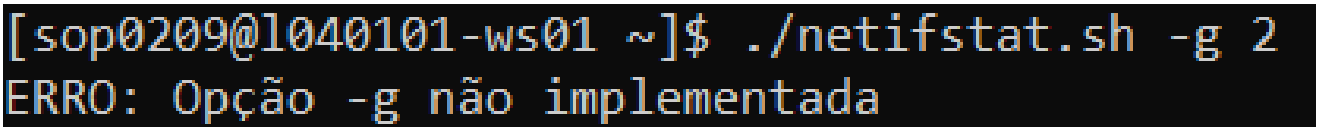

Figura 33 – Erro de inserção de uma opção não implementada no programa

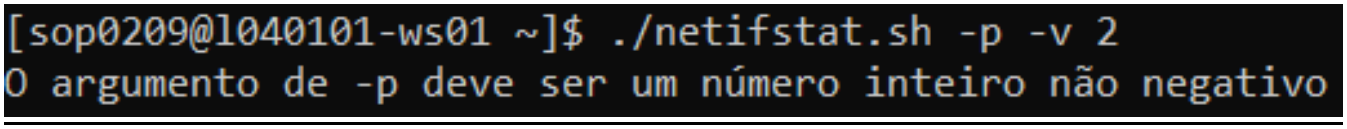

Figura 34 – Erro de inserção de um valor diferente do alertado no erro ou a não inserção de qualquer valor após a opção -p

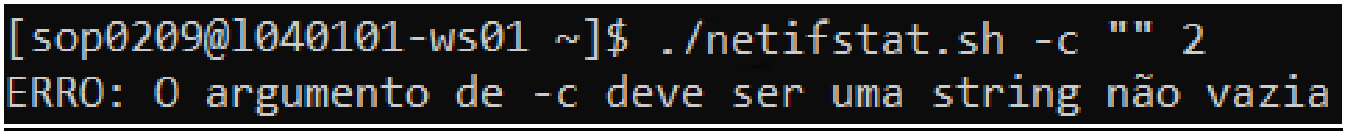

Figura 35 – Erro de inserção de uma string vazia para realizar pesquisa por nome com -c

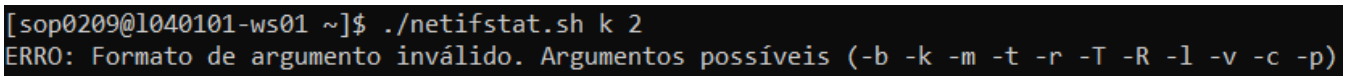

Figura 36 – Erro de inserção de um argumento sem o padrão "-." (caso k)*

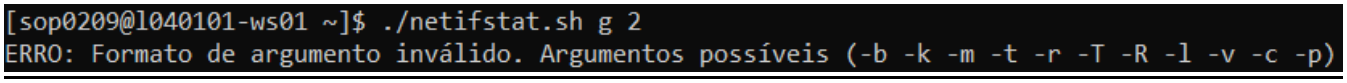

Figura 37 - Erro de inserção de um argumento sem o padrão "-." (caso g)*

Realizado este segundo teste para não ficarmos só com o teste de uma opção que existia mas não estaria no formato certo

Conclusão

O programa netifstat.sh foi implementado corretamente, tendo conseguido produzir as soluções esperadas para todas as opções disponíveis para realizar a sua execução, podendo verificar isto mesmo nos testes realizados.

Este trabalho foi realizado tendo por base os conhecimentos adquiridos nas aulas Teóricas e Práticas da unidade curricular mas essencial e inevitavelmente a maoir parte do conhecimento adquirido necessário para o sucesso do trabalho foi obtido por meio de pesquisa em diversas plataformas que ficaram referenciadas devidamente.

Bibliografia

https://stackoverflow.com/

https://www.geeksforgeeks.org/

https://www.cyberciti.biz/

https://www.redhat.com/

https://www.makeuseof.com/

https://linuxize.com/

https://askubuntu.com/

https://unix.stackexchange.com/

https://www.freecodecamp.org/

https://phoenixnap.com/

https://vitux.com/

https://superuser.com/

https://serverfault.com/

https://linuxhint.com/

https://community.ui.com/

https://www.lifewire.com/