Trabalho realizado por João Gomes Teles Correia e Rafael Remígio Ferreira Lixa Cardoso Coelho.

Neste relatório explicaremos o processo de construção do programa netifstat.sh. Separamos a explicação do programa em 3 partes:

Main (que não é necessariamente uma função mas sim a parte central do programa)

Looping (explica como foi implementada a opção -l passada como argumento)

Functions (explica como foi implementada as funções necessárias)

-Main

Recebemos a informação necessária a partir do comando "ifconfig -a" e da seguinte forma organizamos os dados recebidos:

Criamos 3 arrays um para a informação tx outro para a informação rx e outra para o nome das interfaces a partir do comando ifconfig -a usando grep, awk e tr para selecionar e eliminar as partes recebidas;

Organizamos um array "i\_data" com 3 colunas divididas por um espaço (como uma matriz de n x 3)(e com n igual ao número de interfaces existentes) de modo a serem facilmente percorridas com o awk e guardadas num array com um so String que ajuda nas funções de sorting e desde modo só precisamos de um único array de informação

exemplo:

eth0 0 0

lo 0 0

wifi0 0 0

Depois de criarmos o array "i\_data", usamos o comando sleep

Recebemos a informação de exatamente a mesma forma anterior neste caso para o array "data"

exemplo:

eth0 2 10

lo 6 5

wifi0 7 3

Depois de termos os dois arrays fazemos a subtração dos valores dos RX e TX arrays e salvamos na variável (data) tendo deste modo toda a informação necessária para escrever ou fazer quaisquer cálculos ou sortings intermédios;

Argumentos passados pelo Utilizador:

Iteramos todos os argumentos e usamos um switch case para identificar os seguintes argumentos dando erro se for dado como input algo aqui não especificado

-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

O -b o -k e o -m são opções que mudam o byte\_divisor, uma variável que usamos para dividir os valores de RX e TX (e se tal for o caso TX\_rate e RX\_rate) no final para escrever a data na forma desejada pelo Utilizador

-r Seleciona o uso da função de sorting SortRX (explicada abaixo) através de uma variável booleana

-t Seleciona o uso da função de sorting SortTX (explicada abaixo) através de uma variável booleana

-R Seleciona o uso da função de sorting SortRX (explicada abaixo) através de uma variável booleana pois caso a interface tenha o Rx maior também vai ter o maior R\_Rate

-T Seleciona o uso da função de sorting SortTX (explicada abaixo) através de uma variável booleana pois caso a interface tenha o Tx maior também vai ter o maior T\_Rate

-v Seleciona o uso da função de sorting Reverse (explicada abaixo) através de uma variável booleana

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

Escrever a tabela de valores desejados:

Escrever o cabeçalho:

Dependendo do valor da variável looping, o cabeçalho será

NETIF TX RX TRATE RRATE

ou então em caso de loop

NETIF TX RX TRATE RRATE TXTOT RXTOT

Escrever os valores:

Iteramos pela data criando a variavel TX\_rate e RX\_rate dividindo o tx e o rx pelo tempo de espera, ou seja, o nosso sleep\_time;

Dividimos os valores TX ou RX e as subsequentes somas e rates pelo byte\_divisor que será diferente dependendo da opção selecionada pelo Utilizador

Depois escrevemos a informação no terminal

Exemplo:

eth0 123456 23456 12345.6 2345.6

lo 456 234 45.6 23.4

(ou então no caso de loop)

eth0 123456 23456 12345.6 2345.6 123456 23456

lo 456 234 45.6 23.4 456 234

-Looping

Caso a variável booleana looping seja verdadeira o parte do código de recolhimento de data e de escrita do código da main será iterado com um while loop.

Serão também criados dois outros arrays tot\_tx e tot\_rx onde cada elemento corresponde a uma interface ordenada de forma alfabética sendo somados a ele mesmo da iteração anterior, e depois são inseridos no array sendo depois escritos na nova formatação da tabela

array data:

eth0 2 10 20 3

lo 6 5 6 4

wifi0 7 3 7 10

formatação da escrita no terminal:

NETIF TX RX TRATE RRATE TXTOT RXTOT

eth0 0 0 0 0 0 0

wifi0 0 0 0 0 0 0

-Funções:

SortRX e SortTX

Usamos um algoritmo de ordenação por seleção para organizarmos o array data dos seguintes modos pedido pelo utilizador através da função TX ou RX.

O algoritmo de ordenação por seleção funciona comparando cada elemento i do array com os elementos seguintes posicionando o mais alto no index acima ate acabar o array.

Reverse

Função que inverte o array trocando os valores dos índices com a mesma distância ao centro.

Regex

A função itera por todas as interfaces da data e se o nome da interface corresponder à Expressão Regular desejada esta será adicionada a um array intermedio que depois será passado como o array data;

Alguns testes realizados (Pingando constantemente para google.com para que tenhamos valores e com um sleep time de 10) :

Sem opcões:

-c :

-p :

-b/k/m (m tera valor de 0 pois em 10 segundos não conseguimos obter valores que chegem a 1 mb porém no caso de loop no total a certo ponto vai aparecer) :

-r :

-t :

-R :

-T :

-v (inverso da ordem alfabética neste caso):

-l :

Algumas misturas

-