

**Trabalho Prático 1**

**Taxas de Leitura e Escrita de processos em Bash**

Relatório – Turma P3

Diogo Falcão Nº 108712

José Gameiro Nº108840

**Índice**

[***1.*** ***Introdução*** 3](#_Toc120922000)

[***2.*** ***Desenvolvimento*** 4](#_Toc120922001)

[**2.1** **Inicialização de variáveis e estruturas de dados a utilizar** 4](#_Toc120922002)

[**2.2** **Verificação e validação do argumento obrigatório** 5](#_Toc120922003)

[**2.3** **Obtenção de informações relativas aos processos** 5](#_Toc120922004)

[**2.4** **Argumentos opcionais** 7](#_Toc120922005)

[**2.5** **Impressão de Dados** 10](#_Toc120922006)

[**2.6** **Testes** 11](#_Toc120922007)

[***3.*** ***Conclusão*** 16](#_Toc120922008)

# ***Introdução***

Neste trabalho, foi-nos proposto o desenvolvimento de um script em bash de modo a aplicar os conteúdos lecionados nas aulas práticas e teórico-práticas. O objetivo deste é obter e imprimir estatísticas sobre leitura e escrita de processos de acordo com os argumentos usados na linha de comandos através de uma tabela.

Com isto é possível visualizar o número total de bytes de I/O que um processo leu ou escreveu e também a taxa de leitura ou escrita correspondente ao número de segundos introduzidos pelo utilizador com um parâmetro obrigatório.

Para a impressão dos dados numa tabela existem várias opções que o utilizador pode usar para puder filtrar os dados da tabela, como por exemplo, poder escolher quais os processos que pretende visualizar, ordenar os dados da tabela, entre outros.

Ao longo deste relatório iremos explicar, com detalhe, o processo para a obtenção dos dados relativos aos processos de leitura e escrita, a forma de imprimir os resultados numa tabela e as opções que o utilizador tem ao seu dispor para puder filtrar os processos que deseja visualizar na tabela.

# ***Desenvolvimento***

Nesta parte do relatório, descrevemos qual foi a nossa abordagem para resolver o problema proposto, bem como a explicação dos métodos usados para encontrar uma solução para os diversos problemas que surgiram ao longo da construção deste trabalho, a evolução do código e os testes efetuados para validar a nossa solução

O nosso script está dividido essencialmente em 6 partes:

* Inicialização de variáveis e estruturas de dados a utilizar;
* Verificação e validação do argumento obrigatório;
* Obtenção de informações relativas aos processos;
* Argumentos opcionais;
* Impressão de dados;
* Teste dos dados.

### **Inicialização de variáveis e estruturas de dados a utilizar**

A partir do comando “declare -a”, inicializámos arrays com os tipos de dados correspondentes de cada processo:

* “dates\_seconds” - que irá guardar as datas de início dos processos em segundos;
* “process\_info” - que junta todas estas informações num só array de modo a simplificar futuras operações no script;
* “information” - vai ajudar a fazer a impressão dos dados numa tabela consoante as opções escolhidas;
* “rchar\_array” – que irá guardar os valores de rchar antes de ser executada a função sleep;
* “wchar\_array” – que irá guardar os valores de wchar antes de ser executada a função sleep.

No que se refere a variáveis globais, inicializámos: numProcesses (o número de processos para a opção “-p”, que por defeito é “null”), “write\_values” para sortear os valores de wchar (por defeito esta variável encontra-se com o valor de 0), “reverse” (por defeito é definido com o valor 1, para que a ordenação seja feita por ordem inversa da taxa de leitura).

Como referido anteriormente, existe um parâmetro obrigatório, o número de segundos (a variável “seconds”), que tem de ser colocado como último argumento ao executar o programa.

### **Verificação e validação do argumento obrigatório**

Como já referido anteriormente, o argumento “seconds” trata-se de uma opção obrigatória ao programa que recalcula as estatísticas de leituras e escritas de processos. Dado a importância, este argumento necessita de uma intensiva verificação de modo que o programa corra em consonância.

Ao observar a figura 2 podemos observar que existe uma primeira validação em que, é verificado se o utilizador inseriu pelo menos o tempo de execução. “$#” é uma variável da bash, pré-definida, que guarda o número de argumentos que são introduzidos ao executar o programa, ao verificar se o valor desta variável é 0 estamos a verificar se o utilizador inseriu o tempo de execução. Se este não inseriu, irá aparecer a mensagem presente no comando “echo”, se não avança para a próxima condição.

Na condição seguinte, verifica se o argumento introduzido é um número inteiro positivo através da expressão regular presente na figura 2. Se esta condição for verdadeira irá aparecer no terminal a mensagem de erro presente no comando “echo”, se não continua na execução do código.

### **Obtenção de informações relativas aos processos**

Em primeiro lugar iremos obter o número total de bytes de I/O e de taxa de leitura ou escrita, ou seja, os valores de rchar e wchar, antes de o comando sleep $seconds ser executado para depois com estes valores podermos calcular os valores de rater e ratew. Para isto utilizámos um ciclo for em que este percorre todos os processos e com a condição decisão na linha 41 irá filtrar os processos para aqueles em que temos permissão.

O comando sleep $seconds irá fazer com que o programa espere que passe o número de segundos presente na variável $seconds. Depois, para a obtenção das informações dos processos, implementámos um ciclo for que percorre todos os processos existentes, através do seu PID. Dentro deste ciclo, incluímos três condições de decisão (if statements):

* a primeira para verificar se o processo existe;
* a segunda para verificar se temos permissão para aceder ao processo;
* a terceira e última, para verificar se existe as informações de rchar e wchar.

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

**Fig.1 –** Ciclo for para a indexação dos dados

Se para cada processo, estas três condições forem cumpridas, o programa adiciona ao array process\_info os dados guardados nas variáveis “comm”, “user”, “processID”, “readb”, “writeb”, “rater”, “ratew” e “date”, como podemos observar na imagem abaixo.

A informação “comm” refere-se ao nome do processo, “user” ao utilizador que se encontre registado no computador e “processID” ao ID do processo. As informações do nome do processo e do utilizador obtém-se através das linhas 52 e 53 através de um comando semelhante presente na figura 3 nas linhas 52 e 55. Como o ciclo for utiliza o PID do processo para percorrer todos os processos existentes logo o comando a executar para obter o PID encontra-se na linha 43.

As informações “readb” e “writeb” referem-se ao número total de bytes lidos e escritos, respetivamente. Para as obter utilizámos os comandos nas linhas 61 e 64 e guardámo-las nas variáveis rchar e wchar, respetivamente.

As informações de “rater” e “ratew” referem-se aos valores de taxa de leitura e escrita de processos, respetivamente. Para os obter é necessário subtrair os valores que se encontram nos arrays rchar\_array e wchar\_array aos valores de rchar\_new e wchar\_new, ou seja, os valores depois de executado o comando sleep $seconds. Por fim divide-se o resultado da subtração pelo tempo guardado na variável $seconds e obtém-se assim os valores de rater e ratew onde ficaram guardados nas variáveis rater e ratew.

A data de início do processo é obtida através da linha de comando na linha 92 e é guardada na variável $sDate, que depois irá ser formatada para o formato de mês dia horas:minutos e guardada novamente noutra variável que é a $date. Também é adicionado ao array dates\_seconds um formato da variável $sDate em segundos para depois ser utilizado nas opções -s e -e.

Por fim os valores presentes nas variáveis $comm, $user, $processID, $readb, $writeb, $rater, $ratew e $date são guardados todos no array process\_info pela ordem indicada.

### **Argumentos opcionais**

Para integrar os argumentos opcionais na execução do ficheiro rwstat.sh utilizámos um switch. Existem nove opções e uma opção default. Cada opção está essencialmente dividida em 2 partes: verificação do comando e remoção de toda a informação, que não esteja de acordo com a opção escolhida, dos processos. Para a verificação dos comandos, é usada uma variável, “validate\_args” que conta o número de argumentos inseridos pelo utilizador para cada opção.

Para percorrer todas as opções utilizámos um ciclo while com a função getopts para que se pudesse percorrer todos os argumentos introduzidos pelo utilizador.

* Opção -c

Esta opção tem como objetivo receber uma expressão regular e comparar com os nomes dos comandos presentes no array process\_info para depois puder filtrar e imprimir as informações completas dos processos que estejam de acordo com a expressão regular introduzida pelo utilizador.

A nossa solução para esta opção tem por base guardar o conteúdo da variável $OPTARG (uma variável pré-definida da bash, por causa da função getopts) na variável $comm\_opt. Consequentemente verifica se o utilizador inseriu algum argumento (neste caso, uma expressão regular) após a opção “-c”. Se a condição não for verdadeira então percorre-se o array com a informação de todos os processos e retira-se aqueles cujo o nome do comando não se adequa à expressão regular introduzida (com o comando unset).

* Opção -u

Nesta opção é inserido o nome de um utilizador e, com base nesse nome, é efetuado uma filtração dos dados do array process\_info tal que o nome do utilizador seja igual ao nome do utilizador dos processos.

A solução que encontrámos para este problema foi tal como na opção anterior guardar o valor que se encontra na variável $OPTARG noutra variável chamada $user\_opt, depois fazer uma validação semelhante à da opção -c e se se não verificar nenhum problema é retirado do array process\_info todos os processos que não tenham como utilizador o mesmo que o $user\_opt.

* Opção -s

Esta opção tem o objetivo de dado uma data introduzida pelo utilizador, os processos presentes no array com todas as informações retirar aqueles cuja data de início seja menor do que a data introduzida pelo utilizador.

Semelhante às outras opções o valor de $OPTARG é validado e, se a validação estiver correta, é colocado na variável $min\_date. É efetuada uma segunda validação, para verificar se a data introduzida é válida e se esta for válida são removidos os processos, presentes no array process\_info, que têm uma data de início menor do que a data guardada em $min\_date.

* Opção -e

Esta opção tem um objetivo semelhante ao da opção -s só que as datas dos processos presentes no array com todas as informações que forem maiores do que uma certa data introduzida pelo utilizador serão retiradas.

O algoritmo que encontrámos para esta opção foi semelhante ao da opção -s pois ambas envolvem a filtração dos dados do array process\_info através de uma determinada data, porém neste caso iremos retirar do array process\_info os processos mais as suas informações cujas datas forem maiores do que a data inserida pelo utilizador.

* Opção -m

Nesta opção o utilizador insere um determinado PID e depois irá se retirar os processos do array com todas as informações aqueles que tenham um PID menor do que o inserido.

Para resolvermos este problema, nós fizemos uma validação semelhante às restantes opções e depois na filtração do array, comparámos os PID’s dos processos com o PID guardado na variável $minPID (a variável com o valor do $OPTARG) e retirámos aqueles cujos PID’s fossem menores do o $minPID.

* Opção -M

Esta opção é bastante semelhante à opção -m, no entanto a diferença é que nesta opção são retirados do array com todas as informações os processos que tiverem um PID maior do que um PID introduzido pelo utilizador.

A resolução para este que encontrámos foi, também, muito parecido com a opção -m, em que guardámos o valor de $OPTARG na variável $maxPID, consequentemente, percorremos o array process\_info e retirámos os processos que tinham maior PID do que o guardado em $maxPID.

* Opção -p

Nesta opção o utilizador insere o número de processo que pretende visualizar.

Encontrámos uma solução para este problema em que verificávamos se o utilizador tinha inserido um número, tal como implementamos para as outras opções e verificámos também se o que o utilizador inseriu era um número inteiro positivo. Depois para imprimir o número de processos presente na variável $numProcesses, criámos uma condição, na impressão dos dados, para que esta parasse de imprimir quando tivesse imprimido o número de valores desejados pelo utilizador.

Opção -r e -w

Estas opções são bastantes semelhante e é necessário apenas introduzir -r ou -w ou ambos. Elas servem para ordenar os dados do array com todas as informações dependendo da opção que escolherem. Por defeito os dados encontram-se ordenados por ordem decrescente dos valores de rater, no entanto se a opção escolhida for -w irá os dados por ordem decrescente dos valores de ratew.

Para este problema nós sorteamos os dados de forma a que os valores de rater ficassem por ordem decrescente, sendo esta é a opção default, através da função sort em que selecionámos a coluna que deveria de ordenar (que neste caso é a coluna 6) e por fim a adicionámos uma parte para ordenar array process\_info de acordo com os dados de rater. Caso a opção -w seja introduzida como um argumento a forma de ordenar o array será semelhante à de -r, mudando apenas a coluna a ordenar que neste caso seria a coluna 7.

Opção \*)

Esta opção não é uma que utilizador insere ao correr o programa, mas uma que avisa o utilizador caso este tenha colocado um argumento errado.

* 1. **Impressão de Dados**

Para a impressão de dados, averigua-se em primeiro lugar se o número de processos seja diferente de zero. Depois, se as variáveis globais não mantiverem os mesmos valores e foram alteradas no switch case, isto é, “write\_values” tiver o valor de 1 e “reverse” tiver o valor de 0, então a informação é impressa em conformidade. Se estas variáveis globais não foram mudadas, então o programa vai imprimir todas as informações dos processos por ordem decrescente dos valores de “rater”. Em ambas as situações, quando o número de processos impressos atingir o mesmo valor de “numProcesses”, da opção “-p”, faz break da impressão

Uma imagem com texto

Descrição gerada automaticamente

**Fig.2 –** Impressão de dados

* 1. **Testes**

Durante a realização do projeto, o programa foi alvo de testes básicos, como o argumento obrigatório segundos, até mais complexos, com todos os argumentos opcionais.

* Argumento obrigatório item de existir

Dado que uma das principais funções do programa é calcular, em “s” segundos, a taxa de leitura/escrita (em bytes por segundo) dos processos selecionados nesse intervalo de tempo, o programa dá erro ao não colocar este argumento, isto é, o número de argumentos do programa tem de ser obrigatoriamente igual ou superior a 2.

Uma imagem com texto

Descrição gerada automaticamente

**Fig.3 –** Teste 1

Argumento obrigatório tem de ser um número inteiro positivo.

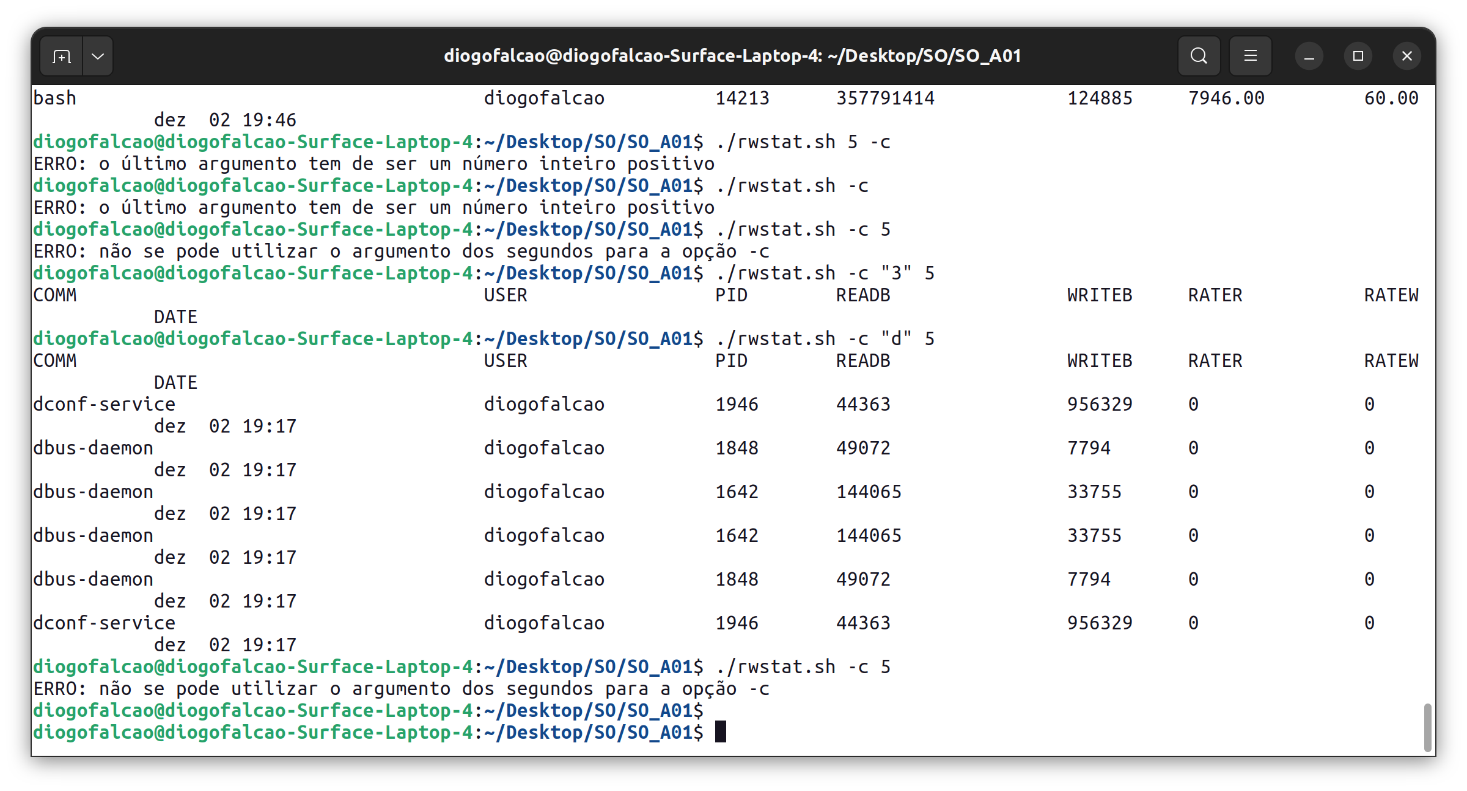
Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

**Fig.4 a) e b) –** Teste 2

* Argumento opcional “–c"

É através da opção “-c” que se pode visualizar no terminal todos os processos que comecem com a expressão regular introduzida. No entanto, para a opção selecionar processos, é necessário introduzir a expressão regular. Assim, se o utilizador apenas colocar no terminal “-c” mais o número de segundos, o programa identifica o erro de que um argumento está em falta.

**Fig.5 –** Teste 3

* Argumento opcional “-u”

Uma imagem com mesa

Descrição gerada automaticamenteAo introduzir “-u” no terminal, é possível filtrar processos cujo nome do utilizador comece com o argumento seguinte escrito no terminal. Mais uma vez, é necessária a introdução de uma parte do nome do utilizador. Assim:

**Fig.6 –** Teste 4

* Argumento opcional “-s”

Este argumento tem como função selecionar uma data mínima. Igualmente às opções anteriores, é necessária a introdução de um argumento, data\_mínima.

Uma imagem com mesa

Descrição gerada automaticamente

**Fig.7 a) –** Teste 5

Para além do utilizador ao usar esta opção ter de escrever o argumento da data\_mínima, esta tem de ser uma data válida.

Uma imagem com mesa

Descrição gerada automaticamente

**Fig.7 b) –** Teste 6

* Argumento opcional “-e”

Uma imagem com texto

Descrição gerada automaticamenteDa mesma forma do argumento “-s”, esta opção seleciona todos os processos que tenham como data máxima o argumento a seguir da opção “-e”, necessitando do argumento seguinte.

**Fig.8 a) –** Teste 7

Uma imagem com texto

Descrição gerada automaticamentePara além do utilizador ao usar esta opção ter de escrever o argumento da data\_máxima, esta tem de ser uma data válida.

**Fig.8 b) –** Teste 8

* Argumento opcional “-m”

Uma imagem com texto

Descrição gerada automaticamenteEsta opção requer um argumento seguinte para a seleção de processos.

**Fig.9 a) –** Teste 9

O argumento seguinte tem de ser um número inteiro positivo.

Uma imagem com mesa

Descrição gerada automaticamente

**Fig.9 b) –** Teste 10

* Argumento opcional “-M”

Uma imagem com texto

Descrição gerada automaticamenteEsta opção requer um argumento seguinte para a seleção de processos.

**Fig.10 a) –** Teste 11

O argumento seguinte tem de ser um número inteiro positivo.

Uma imagem com mesa

Descrição gerada automaticamente

**Fig.10 b) –** Teste 12

* Argumento opcional “-p”

Uma imagem com texto

Descrição gerada automaticamenteFinalmente, a opção “-p” controla o número de processos a imprimir no terminal. Mais uma vez, para o uso desta opção é necessário escrever um argumento que represente o número de processos.

**Fig.11 a) –** Teste 13

Uma imagem com texto

Descrição gerada automaticamente  
Mais uma vez, verifica-se se o argumento introduzido é um número inteiro positivo

**Fig.11 b) –** Teste 14

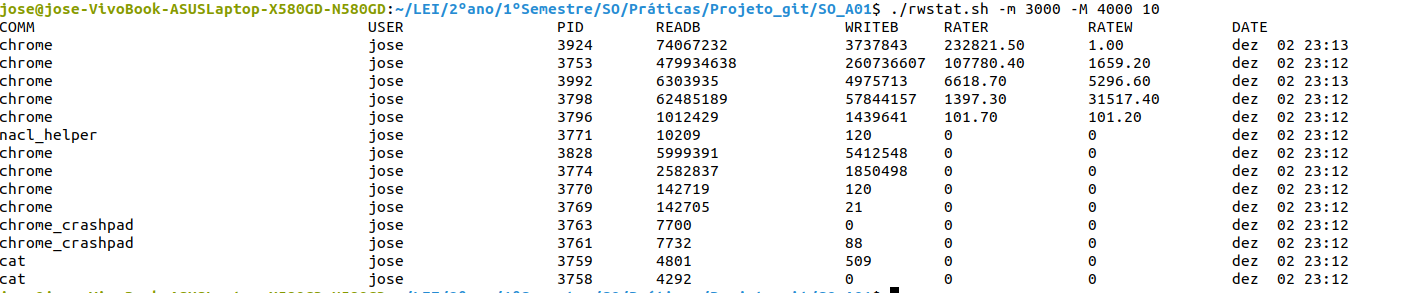
* Argumento “-s” e “-e”

Uma imagem com mesa

Descrição gerada automaticamente Ao utilizar estas opções, verificamos na imagem abaixo, que durante 10 segundos, os processos apresentam uma data maior ou igual à data mínima inserida e menor ou igual à data máxima inserida.

**Fig.12 –** Teste 15

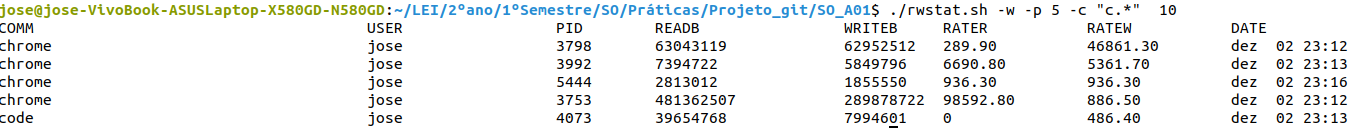
* Argumento “-m” e “-M”

É possível observar na imagem seguinte que ao usar as estas opções e introduzindo no terminal processos com PID’s entre 3000 e 4000 durante 10 segundos, os dados são impressos como esperado.

**Fig.13 –** Teste 16

* Argumento “-w”, “-p” e “-c”

Por fim, ao usarmos uma combinação aleatória de elementos opcionais, de ordenar por ordem os write values, imprimir 5 processos e mostrar todos aqueles que têm o comando começado por “c”, encontramos na imagem seguinte a impressão esperada.



**Fig.14 –** Teste 17

# ***Conclusão***

Resumindo, para este projeto foi desenvolvido um script em bash, apto a obter estatísticas sobre leitura e escrita de processos, de acordo com os argumentos passados pelo utilizador em qualquer combinação. Todos estes argumentos passam por uma série de verificações capazes de parar o programa, caso faltarem os argumentos de opções que o utilizador selecionou. O output do programa será uma tabela com as informações relativas aos processos ocorridos num determinado período de segundos.