deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Trabalho 1 - Gestão de armazenamento

Monitorização do espaço ocupado

Trabalho de Sistemas Operativos
Professor Regente: José Nuno Panelas Nunes Lau

João Monteiro nº114547, João Pinto nº104384

Abstract

O objetivo deste trabalho consistiu no desenvolvimento de dois scripts em bash: *spacecheck.sh* e *spacerate.sh*, os quais permitem a gestão de armazenamento, monitorizando o espaço ocupado em disco por ficheiros com propriedades específicas.

O script *spacecheck.sh* permite visualizar o espaço ocupado pelos ficheiros num diretório fornecido como argumento bem como nos seus subdiretórios. A seleção dos ficheiros pode ser feita de diferentes formas: através de uma expressão regular que é verificada com o nome dos ficheiros (opção -n); através da especificação da data máxima de modificação dos ficheiros (opção -d) ou indicando o tamanho mínimo do ficheiro (opção -s). Além disso, são suportadas opções de ordenação (opções -r, para ordenar por ordem inversa, e -a, para ordenar por nome) e pode também ser limitado o número de linhas da tabela (opção -l). O script trata adequadamente ficheiros e diretórios com espaços no nome, e em casos de inacessibilidade, reporta o espaço ocupado como "NA".

O script *spacerate.sh* compara dois arquivos que contêm a saída do *spacecheck.sh*, exibindo a evolução do espaço ocupado. Este script destaca a diferença entre os espaços ocupados em diretórios presentes em ambos os arquivos e identifica diretórios exclusivos de cada arquivo. As opções de ordenação são similares ao *spacecheck.sh*.

Ambos os scripts permitem uma gestão eficiente do armazenamento ao fornecer informações detalhadas sobre o espaço ocupado em disco e as respetivas variações ao longo do tempo.

Keywords: *Armazenamento; espaço; disco; bytes; scripts; bash; diretórios ; subdiretórios; ficheiros; funções; argumentos.*

1. Espaço ocupado - script *spacecheck.sh*

1.1. Introdução

O script spacecheck.sh desenvolvido permite a visualização do espaço ocupado (em *bytes*) pelos ficheiros selecionados no diretório que lhe é(são) passada(s) como argumento e em todos os subdiretórios destes. O utilizador pode indicar como pretende que a seleção dos ficheiros seja feita. Tal como foi referido no *Abstract* deste documento, a seleção pode ser feita de diferentes formas: através de uma expressão regular que é verificada com o nome dos ficheiros (opção -n); através da especificação da data máxima de modificação dos ficheiros (opção -d) ou indicando o tamanho mínimo do ficheiro (opção -s). Se o utilizador não especificar como deseja que a seleção seja feita todos os ficheiros devem ser contabilizados. Além disso, o utilizador pode escolher como deseja que os dados sejam impressos no ecrã. Os dados podem aparecer ordenados por ordem inversa (opção -r), por ordem alfabética (opção -a) e ainda pode ser escolhido o número de linhas da tabela (opção -l). Apresentamos um exemplo 1 de uma possível interação do utilizador com o script. Neste caso, o utilizador deseja saber o espaço ocupado no diretório Praticas e nos respetivos subdiretórios. A informação aparece ordenada do menor para o maior tamanho pois o utilizador inseriu a opção -r.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -r -n ".*sh" Praticas
SIZE NAME 20231111 -r -n .*sh Praticas
49 Praticas/abc
```

```
260 Praticas/az
492 Praticas/aula1/Teste1/Teste2
921 Praticas/aula2/Teste3/Teste4
956 Praticas/aula1/Teste1
1188 Praticas/aula1
1610 Praticas/aula2/Teste3
2039 Praticas/aula2
3796 Praticas
```

Listing 1: Exemplo de uma possível interação do utilizador com o script

1.2. Explicação do código

O script spacecheck.sh começa com a linha **#!/bin/bash**, ver 2. Esta declaração especifica que o script deve ser interpretado pelo Bash, o interpretador de comandos padrão no ambiente Linux. A variável header é introduzida com a linha **header="$*"**. Aqui, $* representa todos os argumentos passados para o script na linha de comando, e a variável header armazena esses argumentos como uma única string. Esta variável será usada numa fase mais avançada do código. As próximas linhas lidam com opções de seleção e visualização. As variáveis **nome, dataMax** e **tamanhoMin** são inicializadas para armazenar informações relacionadas às opções de seleção. Por exemplo, **dataMax** é inicializada com o valor atual da data e hora, obtido através do comando **date**. As variáveis **r**, **a** e **l** representam opções de visualização e são inicializadas com zero. Estas linhas iniciais do script são fundamentais para configurar variáveis e parâmetros que serão utilizados posteriormente.

```
#!/bin/bash

# Cabecalho
header="$*" # variavel especial que representa como uma unica string todos os argumentos
    passados para o script na linha de comando.

# Opcoes de Selecao
nome=""
dataMax=$(date)
tamanhoMin=""

# Opcoes de Visualizacao
r=0
a=0
l=0
```

Listing 2: Indicação que o script deve ser interpretado pelo Bash. Inicialização das opções de seleção e visualização que serão utilizadas posteriormente.

Em seguida, temos a declaração de um dicionário chamado **space_dict**, ver 3. Neste caso, este dicionário será utilizado para armazenar o tamanho associado a cada diretório. Em seguida, há uma função chamada **display_help**. Esta função é destinada a ser chamada quando nenhum diretório é especificado. Ela imprime uma mensagem de uso do script, explicando as opções disponíveis, e posteriormente encerra o script usando o comando **exit 1**. A função **printHeader** é definida para imprimir um cabeçalho para a saída do script. Ela utiliza a função date para obter a data atual no formato 'YYYYMMDD' e imprime informações como "SIZE", "NAME", a data atual e à frente os argumentos passados pelo utilizador para o script. A função **directory_notFound** é definida para imprimir uma mensagem de erro indicando que um determinado diretório não foi encontrado. Em seguida, o script é encerrado.

```
# Criar o dicionario que vai ser usado para guardar o size associado a cada diretorio.
declare -A space_dict
# Esta funcao e chamada quando nenhum diretorio foi especificado. Explica ao utilizador
    como interagir com o script.
display_help() {
    echo "Usage: $0 [options] directory"
    echo "Options:"
    echo "  -n pattern   Specify a pattern for file names (e.g., '.*sh', '.*pdf', '.*png')"
    echo "  -d date      Specify a maximum date for file modification (format: 'YYYYMMDD')"
    echo "  -s size      Specify a minimum size for files (in bytes)"
    echo "  -r           Sort the output in reverse order"
    echo "  -a           Sort the output by name"
    echo "  -l lines     Limit the number of lines in the table"
    exit 1
}

```

```
# Imprime o cabecalho
printHeader(){
    current_date=$(date +'%Y%m%d')
    printf "%4s %4s %8s %s\n" "SIZE" "NAME" "$current_date" "$*"
}

directory_notFound(){
  echo "ERROR: $1 directory not found!"
  exit 1; # encerra o script
}

printHeader "$header"
```

Listing 3: Declaração do dicionário, estrutura utilizada para guardar a informação solicitada pelo utilizador. Descrição das funções: **display_help**, **printHeader** e **directory_notFound**.

1.2.1. Validação dos argumentos inseridos pelo utilizador

O bloco de código seguinte, ver 4, é responsável por processar as opções fornecidas na linha de comando quando o script é executado. O loop **while** utiliza o comando **getopts** para iterar sobre as opções da linha de comando. Posteriormente, cada opção é tratada pelo bloco **case**, onde são realizadas ações específicas com base na opção fornecida.

```
# Processa as opcoes da linha de comando
while getopts ":n:d:s:ral:" opt; do
    case $opt in # trata cada opcao fornecida na linha de comando
        n)
            nome="$OPTARG"
            if [[ ! "$nome" =~ ^\.\*[^/]+$ ]]; then # expressao regular para verificar se a variavel nome esta no formato desejado
                echo "Invalid pattern for -n. It should be in the format '.*sh', '.*pdf', '.*png', etc."
                exit 1
            fi
            ;;
        d)
            dataMax="$OPTARG"
            if date -d "$dataMax" "+%d %b %H:%M" > /dev/null 2>&1; then
                continue # Se for valida, o script continua para a proxima iteracao do loop
            else
                echo "Input a valid date!"
                exit 1; # Encerra o script com codigo de erro 1
            fi
            ;;
        s)
            tamanhoMin="$OPTARG"
            if ! [[ "$tamanhoMin" =~ ^[0-9]+$ ]]; then
                echo "You have to give a size greater or equal than zero (an integer)!"
                exit 1;
            fi
            ;;
        r)
            r=1
            ;;
        a)
            a=1
            ;;
        l)
            l="$OPTARG"
            if ! [[ "$1" =~ ^[1-9]+$ ]]; then
                echo "You have to give a number of lines greater than zero (an integer)!"
                exit 1;
            fi
            ;;
        \?)
            echo "Invalid option: -$OPTARG" >&2
            exit 1
            ;;
    esac
done
```

Listing 4: Processamento das opções da linha de comando e validação dos argumentos introduzidos.

- **Opção -n (nome):**
  - Esta opção permite definir um padrão para os nomes de ficheiros.
  - A expressão regular `^\.\*[^/]+$` verifica se o padrão está no formato desejado.
  - A expressão regular valida nomes de ficheiros que começam com um ponto, seguido pelo caracter (*), e em seguida, contêm pelo menos um caracter que não seja uma barra ("/").

  Caso o formato especificado não seja respeitado será impressa uma mensagem de erro a explicar como deve ser introduzido o nome e o programa encerra, comando `exit 1`, ver 5.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -n "pdf" Praticas
SIZE NAME 20231111 -n pdf Praticas
Invalid pattern for -n. It should be in the format '.*sh', '.*pdf', '.*png', etc.
```

Listing 5: Opção -n. Necessário respeitar o formato especificado.

- **Opção -d (data):**
  - Permite definir uma data máxima para a última modificação de ficheiros.
  - Utiliza o comando `date -d "$dataMax" "+%d %b %H:%M" > /dev/null 2>&1` para validar a data fornecida. O comando date irá falhar se a data não estiver no formato especificado. Esse formato inclui o dia do mês (%d), a abreviação do mês (%b) - produz a abreviação de três letras do nome do mês em inglês, a hora (%H), e os minutos (%M).
  - `/dev/null 2>&1` é um redirecionamento de saída, descarta a saída padrão e a saída de erro do comando date. Resumidamente: 1) `/dev/null` redireciona a saída padrão (stdout) para o dispositivo nulo (/dev/null) ; 2) `2>&1` redireciona a saída de erro padrão (stderr) para o mesmo local que a saída padrão (stdout).
  - Quando a data for inválida será impressa uma mensagem de erro e o programa encerra, comando `exit 1`.

  Como podemos ver no exemplo 6, quando a data introduzida não corresponde ao formato especificado, é impressa uma mensagem de erro e o programa encerra. Neste exemplo, o utilizador deveria ter inserido "Nov" em vez de "Novemb".

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -d "Novemb 8 10:00"
    Praticas
SIZE NAME 20231111 -d Novemb 8 10:00 Praticas
Input a valid date!
```

Listing 6: Opção -d. A data introduzida deve respeitar o formato especificado.

- **Opção -s (tamanho):**
  - Permite especificar um tamanho mínimo para ficheiros.
  - A validação `if ! [[ "$tamanhoMin" =~ ^[0-9]+$ ]]; then` garante que o tamanho seja um número inteiro maior ou igual a zero. Caso contrário, será impressa uma mensagem de erro e o programa encerrará com o comando `exit 1`.

  Caso seja introduzido um argumento inválido, uma string por exemplo, também é impressa uma mensagem de erro e o programa encerra, ver exemplo 7.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -s "Joao" Praticas
SIZE NAME 20231111 -s Joao Praticas
You have to give a size greater or equal than zero (an integer)!
```

Listing 7: Opção -s. O argumento deve ser um número inteiro maior ou igual a zero.

- **Opções de ordenação: -r e -a:**
  - O utilizador pode escolher como deseja que os dados sejam impressos no ecrã. Os dados podem aparecer ordenados por ordem inversa (opção -r) ou por ordem alfabética (opção -a)
  - Se a opção for introduzida é atribuído o valor **1** à respetiva variável, `r` ou `a`.
- **Opção -l (linhas):**

- Permite definir o número máximo de linhas na tabela de resultados.
- A validação `if ! [[ "$1" =~ ^[1-9]+$ ]]; then` garante que o valor fornecido seja um número inteiro maior que zero.
- Caso contrário, será impressa uma mensagem de erro e o programa encerrará com o comando `exit 1`.

Se não for introduzido nenhum argumento a seguir à opção **l**, é impressa uma mensagem de erro e o programa encerra, ver exemplo 8.

```
joao@joaoh:~/Documents/../SO/SO_Project/Code$ ./spacecheck.sh -l Praticas
SIZE NAME 20231111 -l Praticas
You have to give a number of lines greater than zero (an integer)!

```

Listing 8: Opção -l. Não é inserido nenhum argumento a seguir à opção l.

Caso seja introduzido um argumento inválido, uma string por exemplo, também é impressa uma mensagem de erro e o programa encerra, ver exemplo 9.

```
joao@joaoh:~/Documents/../SO/SO_Project/Code$ ./spacecheck.sh -l "Joao" Praticas
SIZE NAME 20231111 -l Praticas
You have to give a number of lines greater than zero (an integer)!

```

Listing 9: Opção -l. É inserido um argumento inválido.

- **Opção inválida:**
  - `\?` representa uma opção de linha de comando inválida.
  - Será impressa uma mensagem de erro e o programa encerrará com o comando `exit 1`.

Se for introduzida uma opção inválida, por exemplo **-k**, é impressa uma mensagem de erro e o programa encerra, ver 10.

```
joao@joaoh:~/Documents/Universidade/2ano/1S/SO/SO_Project/Code$ ./spacecheck.sh -k
    Praticas
SIZE NAME 20231111 -k Praticas
Invalid option: -k
```

Listing 10: Opção inválida.

1.2.2. Cálculo do espaço ocupado por cada diretório

O código após o ciclo `while` anterior, lida com o processamento dos restantes argumentos introduzidos pelo utilizador, nomeadamente, o(s) diretório(s) a serem processados, ver 11.

```
# Remove as opcoes processadas da linha de comando. Agora, os argumentos restantes em "$@"
    sao os diretorios a serem processados.
shift $((OPTIND-1))

# Verifica se pelo menos 1 diretorio foi especificado
if [ $# -eq 0 ]; then # verifica se restaram argumentos na linha de comando
    echo "ERRO: Nenhum diretorio especificado." >&2
    display_help
fi

# Armazena o diretorio de destino
main_directories=("$@")
```

Listing 11: Código responsável por armazenar em `main_directories` os diretórios a serem processados.

- Após a execução do shift, o array "$@" contém apenas os diretórios fornecidos como argumentos na linha de comando. Os restantes argumentos, as opções de seleção e de visualização deixam de estar em "$@".
- A validação `if [ \$# -eq 0 ]; then` verifica se existem diretórios especificados para serem processados. "$#" é uma variável que representa o número de argumentos na linha de comando. Se não houver diretórios especificados, imprime uma mensagem de erro e exibe a ajuda usando a função `display_help`.

- Os diretórios a serem processados são armazenados em `main_directories`, facilitando assim a manipulação dos diferentes diretórios posteriormente no script.

O seguinte código permite calcular o tamanho de cada diretório, consoante as condições apresentadas pelo utilizador: data máxima, tipo de ficheiro e tamanho máximo de cada ficheiro.

```
# Funcao para calcular o tamanho total de um diretorio e exibi-lo
calculate_directory_size() {
    local dir="$1"
    local total_size="NA"  # Inicializa com "NA"

    # Diretoria nao existe
    [[ -d "$dir" ]] || directory_notFound "$dir"

    local total_size=0 # tem permissao, portanto comeca com SIZE 0

    folders=$(find "$dir" -type d 2>/dev/null)  # alterei
    while IFS= read -r df; do
      total_size=0
      if [[ ! -r "$df" ]] || [[ ! -x "$df" ]] ; then
          space_dict["$df"]="NA"
          continue  # Saltar para o proximo diretorio se nao for acessivel
      fi

      files=$(find "$df" -type f 2>/dev/null)  # Redireciona erros para /dev/null
      while IFS= read -r file; do
          if [[ -f "$file" ]] && [[ "$file" =~ $nome ]] && [[ $(date -r "$file" +%s) -le $(
    date -d "$dataMax" +%s) ]] && [[ $(stat -c %s "$file") -ge "$tamanhoMin" ]]; then
            total_size=$((total_size + $(stat -c %s "$file")))
          fi
      done <<< "$files"
      space_dict["$df"]="$total_size";
    done <<< "$folders"
}
```

- Primeiro começámos por definir duas variáveis: **dir**, com o valor passado no primeiro argumento e também `total_size`, que inicialmente equivale a **"NA"**.
- De seguida, verifica-se se o diretório passado como argumento existe ou não. Caso não exista, a função `directory_notFound` é chamada, passando o diretório como argumento. De seguida, a variável `total_size` passa a 0 e procura-se todas as pastas que têm o diretório passado como argumento. Caso esse *folder* não tenha permissão, adiciona-se ao dicionário `space_dict` esse diretório (como chave) e com valor respetivo "NA".
- Caso contrário percorrem-se todos os ficheiros nesse folder e, consoante as condições apresentadas pelo utilizador, adiciona-se o tamanho desses ficheiros à variável `total_size` ou não. Por fim, adiciona-se ao dicionário a chave que representa o diretório e o seu respetivo valor (tamanho do diretório).

Finalizando este script, temos uma função **display** e ainda um **for loop** que permite percorrer todas as diretorias e calcular o seu tamanho. Por fim, chama-se a própria função **display** para poder reproduzir o seu output na consola.

```
# Funcao para visualizar a ocupacao do espaco como pretendido
display(){
    if [ "$a" -eq 0 ] && [ "$r" -eq 0 ] && [ "$l" -eq 0 ]; then
        for key in "${!space_dict[@]}"; do
            echo "${space_dict[$key]} $key"
        done | sort -r -n -k1
    elif [ "$a" -eq 1 ]; then
        if [ "$r" -eq 1 ]; then
            echo "You can only choose one option between -a and -r. Try again!"
        elif [ "$l" -gt 0 ]; then
            for key in "${!space_dict[@]}"; do
                echo "${space_dict[$key]} $key"
            done | sort -k2 | head -n $l
        else
            for key in "${!space_dict[@]}"; do
                echo "${space_dict[$key]} $key"
            done | sort -k2
        fi
```

```
    elif [ "$r" -eq 1 ]; then
        if [ "$l" -gt 0 ]; then
            for key in "${!space_dict[@]}"; do
                echo "${space_dict[$key]} $key"
            done | sort -n -k1 | head -n $l
        else
            for key in "${!space_dict[@]}"; do
                echo "${space_dict[$key]} $key"
            done | sort -n -k1
        fi
    elif [ "$l" -gt 0 ]; then
        for key in "${!space_dict[@]}"; do
            echo "${space_dict[$key]} $key"
        done | sort -r -n -k1 | head -n $l
    fi
}

for directory in "${main_directories[@]}"; do
    # echo "Processando o diretorio: $directory"
    calculate_directory_size "$directory"
done

display
```

Listing 12: Função display e ciclo for que permite percorrer todas as diretorias e calcular o seu tamanho.

Esta função foi implementada com um **if statement**:

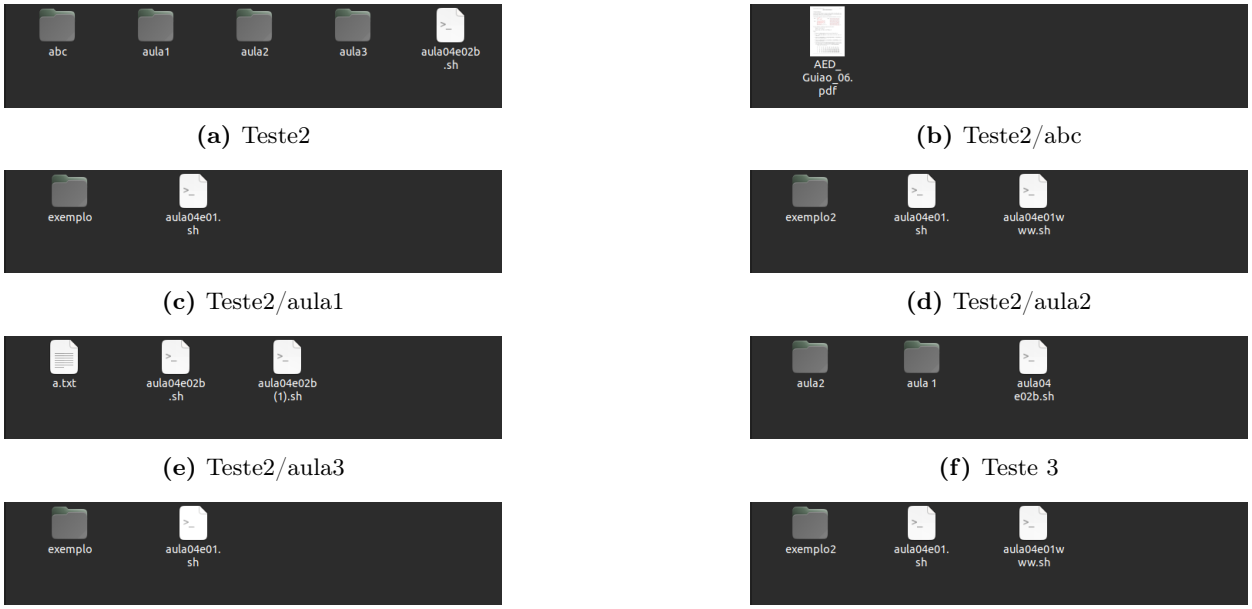
- O primeiro **if**, contempla o cenário em que não é introduzida nenhuma opção de visualização onde o output está ordenado, por *default*, do maior para o menor valor do **SIZE**. Para tal, usámos o "*sort -r -n -k1*" que permite ordenador do maior para o menor valor (-r), de forma numérica (-n), tendo em conta os valores do primeiro campo (-k1);
- De seguida, verificámos os casos em que a variável "**a**" equivale a 1. Caso "**r**" também seja igual a 1, apresentamos uma mensagem de erro, uma vez que apenas se pode escolher uma dessas duas opções. Caso o utilizador tenha transmitido o número de linhas que quer visualizar, então esse número de linhas será imprimido no terminal, ao usar o comando **head**. Tanto nesta ocasião, como na ocasião em que apenas é usada a opção **-a** o output será ordenado por ordem alfabética, devido ao uso do "*sort -k2*", que ordena tendo em conta o segundo campo desse mesmo output.
- Caso a variável "**a**" seja zero e "**r**" esteja com o valor um, o output será ordenado de forma inversa, ou seja tendo em conta o valor de **SIZE**, mas neste caso do menor para o maior. Para tal é usado o comando "*sort -n -k1*" que ordena o primeiro campo do output de forma numérica. Tal como no caso anterior, caso o número de linhas máximo seja especificado, o comando **head** irá entrar em ação para apresentar esse número de linhas.
- Finalmente, caso as variáveis "**a**" e "**r**" sejam iguais a zero e o valor de "**l**" seja maior que zero, o output imprimirá apenas as primeiras **a** linhas do output, que estará na forma *default*, ou seja, ordenado através do valor de **SIZE**, do maior para o menor.
- De referir, que para todos os casos, o output é apresentado da mesma forma: primeiro o valor que está no dicionário `space_dict` e, de seguida, a chave que corresponde a esse valor, de modo ao output ser primeiro a evolução do tamanho e depois o diretório, ou algo mais em casos especiais.

1.3. Validação do script desenvolvido

De seguida, para finalizar o primeiro script, iremos demonstrar os testes feitos ao nosso código, de modo a certificar que tudo funciona como expectável.

Para validar a nossa função é uma boa prática a criação de pastas, com algumas sub-pastas. Assim sendo, seguem o exemplo de duas dessas pastas(**Teste2** e **Teste 3**), que se encontram nos testes desenvolvidos a seguir, que contêm ficheiros de vários tipos, para mostrar a diferença entre vários comandos:

Neste primeiro caso, podemos verificar o **funcionamento da opção -n**, ver 13. É importante referir que como o utilizador não inseriu uma opção de ordenação, os dados aparecem na **ordem default**, do maior para o menor tamanho.



(a) Teste2 **(b)** Teste2/abc

(c) Teste2/aula1 **(d)** Teste2/aula2

(e) Teste2/aula3 **(f)** Teste 3

(g) Teste 3/aula 1 **(h)** Teste 3/aula2

Figure 1. Imagens com os elementos de Teste2 e Teste 3

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -n '.*sh' Teste2
SIZE NAME 20231113 -n .*sh Teste2
2008 Teste2
724 Teste2/aula2
532 Teste2/aula3
492 Teste2/aula1
260 Teste2/aula2/exemplo2
260 Teste2/aula1/exemplo
0 Teste2/abc
```

Listing 13: Desempenho do script *spacecheck* quando o utilizador insere a opção -n.

De seguida, este caso serve simplesmente para confirmar que o programa **não funciona com um diretório não existente**, apresentando a respetiva mensagem de erro, ver 14:

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -n '.*sh' Teste
SIZE NAME 20231113 -n .*sh Teste
ERROR: Teste directory not found!
```

Listing 14: Desempenho do script *spacecheck* quando o utilizador insere um diretório não existente.

Iremos então abranger e confirmar as opções de visualização. Neste caso, a **opção -r funciona** como expectável, os dados aparecem do menor para o maior tamanho (ver 15), contrariamente à **ordem default** apresentada na figura 13.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -r -n '.*sh' Teste2
SIZE NAME 20231113 -r -n .*sh Teste2
0 Teste2/abc
260 Teste2/aula1/exemplo
260 Teste2/aula2/exemplo2
492 Teste2/aula1
532 Teste2/aula3
724 Teste2/aula2
2008 Teste2
```

Listing 15: Desempenho do script *spacecheck* quando o utilizador insere duas opções: a opção -r e a opção -n.

Por outro lado, também a opção **-a** funciona, agora todos os diretórios estão ordenados por ordem alfabética como podemos ver na figura 16.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -a -n '.*sh' Teste2
SIZE NAME 20231113 -a -n .*sh Teste2
2008 Teste2
```

```
0 Teste2/abc
492 Teste2/aula1
260 Teste2/aula1/exemplo
724 Teste2/aula2
260 Teste2/aula2/exemplo2
532 Teste2/aula3
```

Listing 16: Desempenho do script *spacecheck* quando o utilizador insere duas opções: a opção -a e a opção -n.

Em seguida mostramos o funcionamento do script *spacecheck* quando é utilizada a **opção -l** em conjunto com a opção -n. Neste caso, apenas as 2 primeiras linhas são apresentadas no output, ver figura 17. Seguem-se três casos dessa implementação, isolada e também envolvendo outras opções de visualização.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -l 2 -n '.*sh' Teste2
SIZE NAME 20231113 -l 2 -n .*sh Teste2
2008 Teste2
724 Teste2/aula2
```

Listing 17: Desempenho do script *spacecheck* quando o utilizador insere duas opções: a opção -l e a opção -n.

Agora apresentamos o desempenho do script *spacecheck* quando o utilizador insere **três opções: a opção -r, a opção -l e a opção -n.** Uma vez que foi inserida a **a opção -r** os dados são apresentados do menor para o maior tamanho, ver 18.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -r -l 2 -n '.*sh' Teste2
SIZE NAME 20231113 -r -l 2 -n .*sh Teste2
0 Teste2/abc
260 Teste2/aula1/exemplo
```

Listing 18: Desempenho do script *spacecheck* quando o utilizador insere três opções: a opção -r, a opção -l, a opção -n.

Por outro lado o utilizador pode querer ordenar por ordem alfabética, **opção -a**. Na figura 19 apresentamos o desempenho do script *spacecheck* quando o utilizador insere **três opções: a opção -a, a opção -l e a opção -n.**

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -a -l 2 -n '.*sh' Teste2
SIZE NAME 20231113 -a -l 2 -n .*sh Teste2
2008 Teste2
0 Teste2/abc
```

Listing 19: Desempenho do script *spacecheck* quando o utilizador insere três opções: a opção -a, a opção -l, a opção -n.

A partir deste momento, comecámos a verificar se a opção **-d**, referente à data máxima de alteração, funcionava, ver figura 20.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -d "Nov 8 10:00" Teste2
SIZE NAME 20231113 -d Nov 8 10:00 Teste2
251962 Teste2
250220 Teste2/abc
724 Teste2/aula2
492 Teste2/aula1
266 Teste2/aula3
260 Teste2/aula2/exemplo2
260 Teste2/aula1/exemplo
```

Listing 20: Desempenho do script *spacecheck* quando o utilizador insere a opção -d

Atentando nos próximos dois casos, reparamos que entre 3 de Novembro e 11 do mesmo mês, o tamanho do diretório em questão difere, uma vez que existe um ficheiro que foi, pela última vez modificado a dia 10 de Novembro.

Portanto quando consultamos o tamanho do diretório até à data 3 de Novembro, ver Figura 21, constatamos que tem 266 bytes.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -d "Nov 3 10:00" Teste2/
    aula3
SIZE NAME 20231113 -d Nov 3 10:00 Teste2/aula3
266 Teste2/aula3
```

Listing 21: Visualização do espaço ocupado até 3 de Novembro.

No entanto, ao consultar o tamanho do diretório até à data 11 de Novembro já constatamos que tem um valor superior, 538 bytes, ver figura 22. Isto indica que existe um ficheiro que foi, pela última vez modificado a entre 3 e 11 de novembro.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -d "Nov 11 10:00" Teste2/
    aula3
SIZE NAME 20231113 -d Nov 11 10:00 Teste2/aula3
538 Teste2/aula3
```

Listing 22: Visualização do espaço ocupado até 11 de Novembro.

A **opção -d** pode ser utilizada combinada com outras opções. Em baixo apresentamos o desempenho do script *spacecheck* quando o utilizador insere a opção -d com a opção -l (ver figura 23) e com a opção -r (ver figura 24). É importante referir que como o nome dos ficheiros não é especificado, todos os tipos de ficheiros devem ser contabilizados.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -d "Nov 8 10:00" -l 2 Teste2
SIZE NAME 20231113 -d Nov 8 10:00 -l 2 Teste2
251962 Teste2
250220 Teste2/abc
```

Listing 23: Desempenho do script *spacecheck* quando o utilizador insere **duas opções**: a opção -d e a opção -l.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -r -d "Nov 8 10:00" -l 2
    Teste2
SIZE NAME 20231113 -r -d Nov 8 10:00 -l 2 Teste2
260 Teste2/aula1/exemplo
260 Teste2/aula2/exemplo2
```

Listing 24: Desempenho do script *spacecheck* quando o utilizador insere **três opções**: a opção -r, a opção -d e a opção -l.

Tratando a última das opções de seleção, confirmamos que também a opção **-s**, funciona corretamente. Neste caso, o utilizador especifica o tamanho minímo dos ficheiros, ver figura 25.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -s 240 Teste2/aula2
SIZE NAME 20231113 -s 240 Teste2/aula2
260 Teste2/aula2/exemplo2
260 Teste2/aula2
```

Listing 25: Desempenho do script *spacecheck* quando o utilizador insere a opção -s.

Nos dois casos seguintes, podemos verificar a diferença do uso da opção **-s**, usando como argumento o mesmo diretório (**Teste2/aula1**). Na figura 26 encontra-se o output quando o utilizador especificou um tamanho mínimo de 240 bytes.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -s 240 Teste2/aula1
SIZE NAME 20231113 -s 240 Teste2/aula1
260 Teste2/aula1/exemplo
260 Teste2/aula1
```

Listing 26: Desempenho do script *spacecheck* quando o utilizador insere a opção -s especificando um **tamanho mínimo de 240 bytes**.

Na figura 27 encontra-se o output quando o utilizador não insere restrições relativamente ao tamanho mínimo dos ficheiros. Neste caso, o folder **Teste2/aula1** tem um size maior associado do que tinha anteriormente na figura 26, o que indica que este folder tem pelo menos um ficheiro com tamanho inferior a 240 bytes.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh Teste2/aula1
SIZE NAME 20231113 Teste2/aula1
492 Teste2/aula1
260 Teste2/aula1/exemplo
```

Listing 27: Desempenho do script *spacecheck* quando o utilizador não insere restrições relativamente ao tamanho mínimo dos ficheiros.

Quando não é possível aceder a uma diretoria ou determinar o tamanho de um ficheiro numa diretoria, o espaço ocupado pelos ficheiros dessa diretoria deve ser assinalado com **NA**. Na figura 28 demonstramos o funcionamento do nosso script quando o utilizador insere um diretório (neste exemplo: **/etc**) que tem subdiretórios que o utilizador não tem permissão. Como podemos constatar o seu tamanho aparece assinalado com **NA**.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh /etc
SIZE NAME 20231113 /etc
(...)
31 /etc/depmod.d
28 /etc/dconf/profile
27 /etc/brltty/Input/no
24 /etc/insserv.conf.d
18 /etc/gdm3/PostSession
NA /etc/ssl/private
NA /etc/polkit-1/localauthority
NA /etc/libvirt/secrets
NA /etc/cups/ssl
0 /etc/xdg/systemd
0 /etc/X11/xorg.conf.d
0 /etc/X11/xkb
(...)
```

Listing 28: Desempenho do script *spacecheck* quando o utilizador não tem permissão para determinados diretórios.

Também os **diretórios que contêm espaços** (neste caso: **Teste 3**) são capazes de serem interpretados pelo script, como o exemplo 29 demonstra.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -n ".*sh" Teste\ 3
SIZE NAME 20231113 -n .*sh Teste 3
1476 Teste 3
724 Teste 3/aula2
492 Teste 3/aula 1
260 Teste 3/aula2/exemplo2
260 Teste 3/aula 1/exemplo
```

Listing 29: Desempenho do script *spacecheck* quando o utilizador insere diretórios com espaços no nome.

Por último, é também **possível passar mais do que um diretório como argumento** como podemos ver na figura 30. O processamento dos diferentes diretórios é possível devido ao **ciclo for** presente na figura 12, como foi visto anteriormente.

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacecheck.sh -s 100 Teste2 Teste2/abc
    Teste\ 3
SIZE NAME 20231113 -s 100 Teste2 Teste2/abc Teste 3
252228 Teste2
250220 Teste2/abc
1476 Teste 3
724 Teste 3/aula2
724 Teste2/aula2
532 Teste2/aula3
492 Teste 3/aula 1
492 Teste2/aula1
260 Teste 3/aula2/exemplo2
260 Teste 3/aula 1/exemplo
260 Teste2/aula2/exemplo2
260 Teste2/aula1/exemplo
```

Listing 30: Desempenho do script *spacecheck* quando o utilizador insere mais do que um diretório como argumento.

2. Evolução do espaço ocupado - script *spacerate.sh*

2.1. Introdução

O segundo script criado, **spacerate.sh**, permite monitorizar a evolução do espaço ocupado em diferentes diretorias. Para tal compara dois ficheiros que resultam da execução do primeiro script, spacecheck.sh, onde ao ser executado esse script, o output resultante é redirecionado para um ficheiro. Após a criação desses ficheiros, podemos finalmente executar o spacerate.sh, com dois desses ficheiros passados como argumentos. Para além desses dois ficheiros, existe ainda a possibilidade de passar como argumentos opções de visualização idênticas às do primeiro script (opções -a, -l e -r). Durante o relatório existem explicações de como estas opções funcionam, e as alterações que produzem no output. Falando especificamente do script, ele observa os ficheiros, onde estão especificadas diretorias. Posteriormente, nas diretorias que estão especificadas em ambos os ficheiros, calcula-se e apresenta-se a evolução do espaço ocupado por essas diretorias. Para aquelas que não se encontram num dos ficheiros, por outro lado, haverá uma apresentação especial ao utilizador,

utilizando as palavras **NEW** ou **REMOVED**. Apresentamos então um simples exemplo de uma interação do utilizador com este script:

```
joao@joaoh:~/Documents/.../SO/SO_Project/Code$ ./spacerate.sh spacecheck_20231123
    spacecheck_20221123
SIZE NAME
30 Teste2
15 Teste2/aula4/ex1 NEW
10 Teste2/aula4/ex2 NEW
0 Teste2/aula3
-5 Teste2/aula4
```

2.2. Explicação do código

Este script começa com a inicialização das opções de visualização, tal e qual como no script spacecheck.sh, e, para além disso, possui uma simples função para imprimir o *header*, **printHeader**, como pretendido.

```
#!/bin/bash

# Opcoes de Visualizacao
r=0
a=0
l=0

# Da print ao cabecalho
printHeader(){
    printf "%4s %4s\n" "SIZE" "NAME"
}
```

Posteriormente, através de um ciclo *while*, fazendo uso, mais uma vez, do comando *getopts*, percorremos as opções fornecidas pelo utilizador na linha de comando, onde se validam os argumentos fornecidos, e onde se atribuem valores às variáveis de visualização.

```
# Processa as opcoes da linha de comando
while getopts ":ral:" opt; do
    case $opt in
        r)
            r=1
            ;;
        a)
            a=1
            ;;
        l)
            l="$OPTARG"
            if ! [[ "$l" =~ ^[1-9]+$ ]]; then
                echo "You have to give a number of lines greater then zero (an integer)!"
                exit 1;
            fi
            ;;

        \?)
            echo "Invalid option: -$OPTARG" >&2
            exit 1
            ;;
    esac
done
```

Uma vez que cada uma destas opções já foi apresentada anteriormente apresentamos em baixo algumas das possíveis mensagens de erro que podem aparecer ao utilizador.

- **Opção -l (linhas):**

  Ao inserir a opção **l** é necessário passar um argumento. Caso contrário é impressa uma mensagem de erro e o programa encerra.

  ```
  joao@joaoh:~/Documents/../SO/SO_Project/Code$ ./spacerate.sh -l spacecheck_20231123
      spacecheck_20221123
  You have to give a number of lines greater then zero (an integer)!

  ```

  Também ao introduzir um argumento inválido, como um número não natural, é impressa uma mensagem de erro e o programa encerra.

```
joao@joaoh:~/Documents/../SO/SO_Project/Code$ ./spacerate.sh -l 2.5
    spacecheck_20231123 spacecheck_20221123
You have to give a number of lines greater then zero (an integer)!

```

- **Opção inválida:** Ao introduzir uma opção inválida, por exemplo **-t**, é impressa uma mensagem de erro e o programa encerra.

```
joao@joaoh:~/Documents/Universidade/2ano/1S/SO/SO_Project/Code$./spacerate.sh -t
    spacecheck_20231123 spacecheck_20221123
Invalid option: -t
```

Após este bloco de código, o seguinte irá processar os argumentos que restam e inicializa três dicionários e atribui valores a duas novas variáveis:

```
# Remove as opcoes processadas da linha de comando
shift $((OPTIND-1))

# Agora, os argumentos remanescentes sao os ficheiros

if [ ! "$#" -eq 2 ]; then # verifica se restaram apenas dois argumentos na linha de comando
    echo "ERRO: Especifique dois ficheiros."
    exit 1;
fi


fileA="$1"
fileB="$2"
declare -A fileA_dict
declare -A fileB_dict
declare -A rate_dict
```

- Mais uma vez, a execução do shift, irá permitir a remoção dos argumentos que já foram processados. Sendo assim, apenas é expectável que sobrem os ficheiros no array "$@".
- De seguida, o **if statement** procura verificar se restam exatamente dois argumentos que, neste caso, representam ambos os ficheiros a ser tratados. Graças a essa verificação, pode-se atribuir esses dois argumentos às variáveis **"fileA"** e **"fileB"**.
- Por último criam-se três dicionários, para processar a informação presente nos ficheiros.

O seguinte bloco de código apresenta uma função que calcula o tamanho real de um diretório, o que permite o cálculo da real evolução do espaço ao longo tempo:

```
realSize(){
    local directory="$1"
    local size=0

    if [[ "$2" == "A" ]]; then
      for key in "${!fileA_dict[@]}"; do
          if [[ "$key" == "$directory" || ("$key" == "$directory"/* && "$key" != "
    $directory"/*/*) ]]; then
              if [[ "$key" == "$directory" ]]; then # Se for o proprio diretorio
                size=$((size + fileA_dict["$key"]))
              else
                  size=$((size - fileA_dict["$key"])) # A chave e um subdiretorio direto do
     diretorio pai. Caso contrario estaria a descontar mais do que uma vez (se considerasse
     apenas "$directory"/*).
              fi
          fi
      done
    else
      for key in "${!fileB_dict[@]}"; do
          if [[ "$key" == "$directory" || ("$key" == "$directory"/* && "$key" != "
    $directory"/*/*) ]]; then
              if [[ "$key" == "$directory" ]]; then # Se for o proprio diretorio
                size=$((size + fileB_dict["$key"]))
              else
                  size=$((size - fileB_dict["$key"])) # A chave e um subdiretorio direto do
     diretorio pai. Caso contrario estaria a descontar mais do que uma vez (se considerasse
     apenas "$directory"/*).
```

```
            fi
        fi
    done
fi

echo "$size"
}
```

- Para começar inicializam-se duas variáveis; **directory** equivale ao primeiro argumento passado ao chamar esta função; **size** é inicializada com valor igual a zero.
- De seguida, verifica se o segundo argumento passado equivale a "A" ou a "B". Assim sendo, vamos indicar com que ficheiro estamos a trabalhar. O processo é igual para ambos os casos, por isso a explicação vai ser global.
- Através do uso de um **for loop**, percorre-se todas as chaves que se encontram nos dicionários. Caso essa chave seja igual ao argumento passado ou essa chave corresponda a um subdiretório direto do diretório passado como argumento o primeiro "if statement" é válido. Em seguida, verifica-se se a chave é especificamente igual ao argumento. Caso isso se verifique, incrementa-se à variável **size** o valor correspondente a essa chave. No caso contrário, retira-se ao valor de **size** o valor dessa chave. Resumindo, o tamanho de um diretório será igual ao seu tamanho total subtraído pelos valores dos tamanhos dos seus diretórios descendentes.

De seguida temos presente a função fundamental deste script (**calculate_size_evolution**), uma vez que é esta que nos permite calcular a evolucao do espaco ocupado e construir o dicionário **rate_dict** que contém os dados que serão impressos (evolução do espaço associado a cada diretório).

```
# Funcao que calcula a evolucao do espaco ocupado
calculate_size_evolution(){
    while read -r size path; do
        if [[ "$size" == "SIZE" ]]; then
            continue
        else
            fileA_dict["$path"]="$size"

        fi
    done <<< "$(cat "$fileA" | awk '{ printf $1; for (i=2; i<=NF; i++) printf " %s", $i; print "" }')"

    while read -r size path; do
            if [[ "$size" == "SIZE" ]]; then
                continue
            else
                fileB_dict["$path"]="$size"

            fi
    done <<< "$(cat "$fileB" | awk '{ printf $1; for (i=2; i<=NF; i++) printf " %s", $i; print "" }')"

    # Loop para calcular as diferencas
    for key in "${!fileA_dict[@]}"; do
        if [[ -n ${fileB_dict[$key]+x} ]]; then
            sizeB=$(realSize "$key")
            sizeA=$(realSize "$key" "A")
            size_rate=$((sizeA-sizeB))
            rate_dict["$key"]=$size_rate
        else
            sizeA=$(realSize "$key" "A")
            key_new="$key NEW"
            rate_dict["$key_new"]=$sizeA
        fi
    done

    for key in "${!fileB_dict[@]}"; do
        if [[ ! -n ${fileA_dict[$key]+x} ]]; then
            sizeB=$(realSize "$key")
            key_removed="$key REMOVED"
            rate_dict["$key_removed"]=-$sizeB
        fi
    done
```

```
42
43 }
```

- Inicialmente, para cada ficheiro usa-se um **while loop**, para percorrer cada linha dos dois ficheiros e dividímos essas linhas em duas partes, *"size" e "path"*, que correspondem, respetivamente, ao tamanho do diretório e o próprio diretório.
- Caso o *"size"* seja igual a **"SIZE"**, ignora-se essa linha, uma vez que é a linha do cabeçalho (os ficheiros começam por norma com SIZE NAME ...). Caso contrário, adiciona-se aos respetivos dicionários (**fileA_dict** e **fileB_dict**) os valores lidos do ficheiro (graças ao comando **cat**), onde o *path* será a chave do dicionário e o respetivo *size* o valor que corresponde a essa chave. O comando **awk**, permite formatar a linha, de modo a ignorar espaços desnecessários.
- Terminada essa parte, o **for loop** permite iterar por cada chave presente no dicionário A. Se essa chave também estiver presente no dicionário B, calcula-se o tamanho real para esse diretório. Na variável `size_rate` vai-se guardar o valor do tamanho real no ficheiro A menos o valor do tamanho real no ficheiro B. Posto isto, guarda-se num novo dicionário (`rate_dict`) essa chave (que representa um diretório) com o valor que está na variável `size_rate`.
- Caso essa chave não esteja no dicionário B, então calcula-se o valor real desse diretório no ficheiro correspondente e a chave vai alocar também a string **"NEW"**, para definir que este é um novo diretório, dada a evolução temporal. A nova chave vai ser guardado no dicionário que mostra a evolução e o valor correspondente vai ser, simplesmente, o valor alocado na variável **sizeA**.
- Por último, percorre-se as chaves no dicionário B e, caso alguma não esteja contida no dicionário A, o processo será similar ao descrevido no último ponto. No entanto, neste caso a chave passará a alocar a string **"REMOVED"** e o valor correspondente à nova chave será o inverso do valor que está armazenado na variável **sizeB**.

Para finalizar este script, utilizámos uma função display, para fornecer o output esperado ao utilizador, ver figura 31. A explicação para esta função, foi já dada num momento anterior a este, ver 12. De modo que o código opere como esperado é feita no final do script **spacerate.sh** a chamada a três funções anteriormente explicadas: `calculate_size_evolution`, **printHeader** e **display**.

```
1  # Funcao para visualizar a ocupacao do espaco como pretendido
2  display(){
3      if [ "$a" -eq 0 ] && [ "$r" -eq 0 ] && [ "$l" -eq 0 ]; then
4          for key in "${!rate_dict[@]}"; do
5              echo "${rate_dict[$key]} $key"
6          done | sort -r -n -k1
7      elif [ "$a" -eq 1 ]; then
8          if [ "$r" -eq 1 ]; then
9              echo "You can only choose one option between -a and -r. Try again!"
10         elif [ "$l" -gt 0 ]; then
11             for key in "${!rate_dict[@]}"; do
12                 echo "${rate_dict[$key]} $key"
13             done | sort -k2 | head -n $l
14         else
15             for key in "${!rate_dict[@]}"; do
16                 echo "${rate_dict[$key]} $key"
17             done | sort -k2
18         fi
19     elif [ "$r" -eq 1 ]; then
20         if [ "$l" -gt 0 ]; then
21             for key in "${!rate_dict[@]}"; do
22                 echo "${rate_dict[$key]} $key"
23             done | sort -n -k1 | head -n $l
24         else
25             for key in "${!rate_dict[@]}"; do
26                 echo "${rate_dict[$key]} $key"
27             done | sort -n -k1
28         fi
29     elif [ "$l" -gt 0 ]; then
30         for key in "${!rate_dict[@]}"; do
31             echo "${rate_dict[$key]} $key"
32         done | sort -r -n -k1 | head -n $l
33     fi
34 }
```

```
35
36 calculate_size_evolution
37 printHeader
38 display
```

Listing 31: Função display e chamada a funções.

2.3. Validação do script desenvolvido

Para finalizar, iremos mostrar alguns outputs resultantes do script **spacerate.sh**. De forma a aferirmos a validade e qualidade do script desenvolvido utilizámos dois ficheiros de teste: **spacecheck_2022** e **spacecheck_2023**, cujo conteúdo mostramos nas figuras 32 e 33 respetivamente.

```
1 SIZE NAME 20231108 fileB
2 70 Teste 2
3 40 Teste 2/abc
4 40 Teste 2/abc/1
5 30 Teste 2/aula2
6 10 Teste 2/aula2/1
7 5 Teste 2/aula2/lixo
8 15 Teste3
```

Listing 32: Conteúdo do ficheiro spacecheck_2022

```
1 SIZE NAME 20231108 fileA
2 50 Teste 2
3 20 Teste 2/abc
4 10 Teste 2/abc/1
5 10 Teste 2/abc/2
6 30 Teste 2/aula2
7 15 Teste 2/aula2/1
8 10 Teste 2/aula2/2
9 5 Teste 2/aula2/3
10 10 Teste3
```

Listing 33: Conteúdo do ficheiro spacecheck_2023

Apresentamos um exemplo simples da implementação deste script, sem opções de visualização, onde podemos verificar a evolução do espaço ao longo do tempo e a definição especial ("NEW" ou "REMOVED") de certos diretórios.

```
1 joao@joaoh:~/.../SO/SO_Project/Code$ ./spacerate.sh spacecheck_2023 spacecheck_2022
2 SIZE NAME
3 10 Teste 2/aula2/2 NEW
4 10 Teste 2/abc/2 NEW
5 5 Teste 2/aula2/3 NEW
6 5 Teste 2/aula2/1
7 0 Teste 2
8 0 Teste 2/abc
9 -5 Teste3
10 -5 Teste 2/aula2/lixo REMOVED
11 -15 Teste 2/aula2
12 -30 Teste 2/abc/1
```

Nos seguintes exemplos, vemos o uso correto das várias opções de visualização. Primeiro a opção **-l**, que funciona como esperado:

```
1 joao@joaoh:~/.../SO/SO_Project/Code$ ./spacerate.sh -l 2 spacecheck_2023 spacecheck_2022
2 SIZE NAME
3 10 Teste 2/aula2/2 NEW
4 10 Teste 2/abc/2 NEW
```

De seguida, a opção **-r**, que apresenta os valores do campo **"SIZE"**, ordenados do menor para o maior:

```
1 joao@joaoh:~/.../SO/SO_Project/Code$ ./spacerate.sh -r spacecheck_2023 spacecheck_2022
2 SIZE NAME
3 -30 Teste 2/abc/1
4 -15 Teste 2/aula2
5 -5 Teste 2/aula2/lixo REMOVED
6 -5 Teste3
7 0 Teste 2/abc
8 0 Teste 2
```

```
5 Teste 2/aula2/1
5 Teste 2/aula2/3 NEW
10 Teste 2/abc/2 NEW
10 Teste 2/aula2/2 NEW
```

Por último, o output ordenado pela ordem alfabética dos diretórios:

```
joao@joaoh:~/.../SO/SO_Project/Code$ ./spacerate.sh -a spacecheck_2023 spacecheck_2022
SIZE NAME
0 Teste 2
0 Teste 2/abc
-30 Teste 2/abc/1
10 Teste 2/abc/2 NEW
-15 Teste 2/aula2
5 Teste 2/aula2/1
10 Teste 2/aula2/2 NEW
5 Teste 2/aula2/3 NEW
-5 Teste 2/aula2/lixo REMOVED
-5 Teste3
```

Nestes últimos dois exemplos, demonstramos o uso de mais do que uma opção de visualização.

```
joao@joaoh:~/.../SO/SO_Project/Code$ ./spacerate.sh -a -l 2 spacecheck_2023 spacecheck_2022
SIZE NAME
0 Teste 2
0 Teste 2/abc
```

```
joao@joaoh:~/.../SO/SO_Project/Code$ ./spacerate.sh -r -l 2 spacecheck_2023 spacecheck_2022
SIZE NAME
-30 Teste 2/abc/1
-15 Teste 2/aula2
```

3. Conclusões

Este trabalho teve como objetivo o desenvolvimento de dois scripts que, como já foi referido, permitem fazer um cálculo do espaço ocupado por diferentes diretorias e a visualização desse mesmo espaço e, também, verificar a evolução dessa mesma ocupação de espaço através de ficheiros. Para além desse mesmo desenvolvimento, este trabalho também nos habilitou, de certo modo, a ir mais além dentro da programação em Bash, uma vez que nos permitiu desenvolver conhecimentos nessa área e, posteriormente, pôr em prática essas aptidões.

References

[1] Intro to Bash Regular Expressions. Site consultado a 27/10/2023: https://dev.to/zachgoll/intro-to-bash-regular-expressions-4d2p.

[2] How to use $OPTARG on getopts? Site consultado a 31/10/2023: https://stackoverflow.com/.

[3] How to Use a Key Value Dictionary in Bash? Site consultado a 05/11/2023: https://linuxhint.com/use-key-value-dictionary-bash/.

[4] Comando man para perceber a funcionalidade de diversos comandos.

[5] Comando date: Site consultado a 24/10/2023: https://www.geeksforgeeks.org/date-command-linux-examples/.