Gestão de armazenamento

MONITORIZAÇÃO DO ESPAÇO OCUPADO

Abel José Enes Teixeira 113655| Diogo Lopes Oliveira 113664

11/11/2023

Sistemas Operativos

Prof. António Guilherme Rocha Campos

Prof. **Regente** José Nuno Panelas Nunes Lau

Ano letivo

2023/2024

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

Gestão de armazenamento

ÍNDICE

Gestão de armazenamento

INTRODUÇÃO

O trabalho sobre o qual incidirá este relatório visa o desenvolvimento de scripts em ***bash*** que permitem monitorizar o espaço ocupado em disco por ficheiros com determinadas propriedades.

Deste trabalho resultou a criação de ferramentas que permitem uma melhor gestão do armazenamento, como por exemplo, a possibilidade de verificar a variação do espaço ocupado em disco, espaço total ocupado, entre outras funcionalidades.

Ao longo deste relatório, serão evidenciados os problemas que foram surgindo ao longo do processo de elaboração do código, acompanhados da sua respetiva resolução, com a posterior validação.

O relatório conta com várias *ScreenShots* que auxiliam a compreensão dos obstáculos, tanto do *Visual Studio Code*, que foi o editor de código usado, como do terminal do *Linux* no qual eram impressos os resultados do código.

METODOLOGIA

O trabalho começou com uma leitura minuciosa do exercício pedido e, acima de tudo, com uma análise do resultado expectável após correr o programa, pois foi este o nosso ponto de referência.

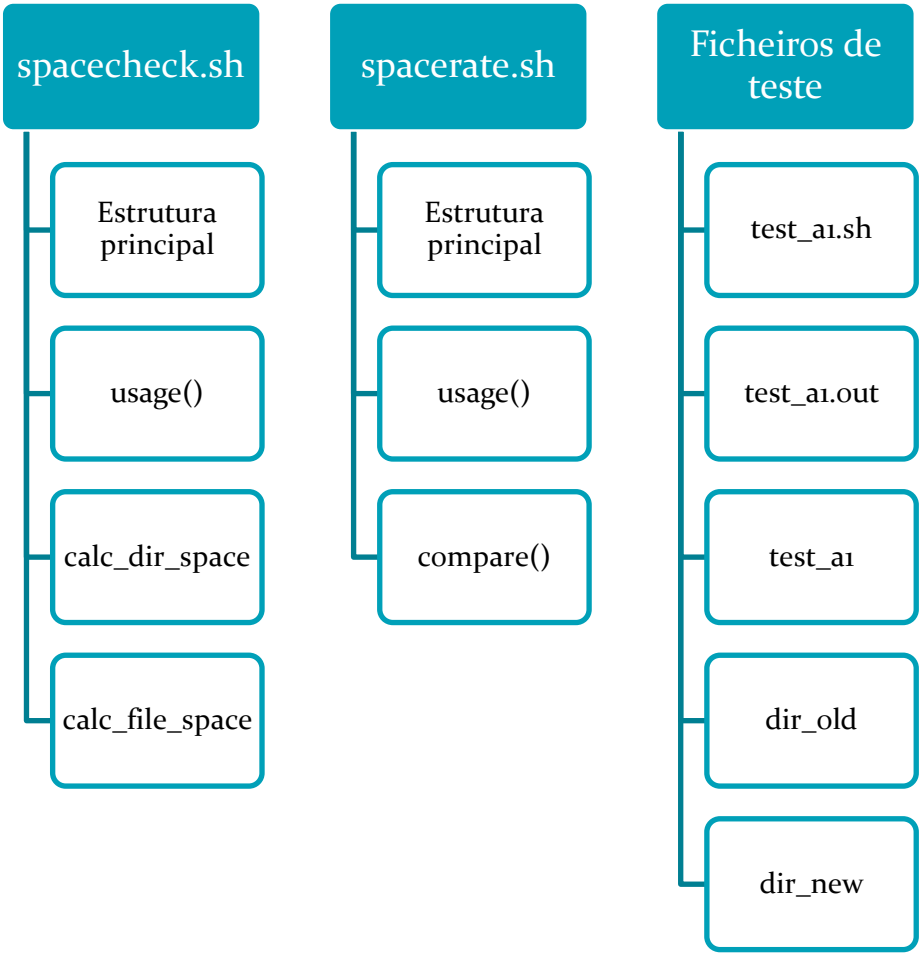
O início foi bastante lento, pois ao início não era claro qual o objetivo proposto, no entanto, à medida que fomos elaborando o código, acabou por se tornar mais simples do que inicialmente parecia.

Para o *spacecheck.sh*, decidimos iniciar a escrita do código e desde logo, dar prioridade à sua organização para facilitar a resolução de erros que certamente iriam surgir mais tarde. Essa organização foi suportada pela implementação de funções que dão *"return"* de informações essenciais para a concretização do trabalho.

No fim, acabou por se tornar uma mais valia, pois facilitou imenso a compreensão do código e a deteção de erros que foram aparecendo, o que nos levou a adotar a mesma metodologia para a segunda parte, *spacerate.sh* .

Gestão de armazenamento

Arquitetura do programa



- **Spacecheck.sh**
  - Estrutura Principal – código não implementado em funções
  - Funções:
    - calc_dir_space()
    - calc_file_space()
    - usage()

Gestão de armazenamento

- **Spacerate.sh**
  - Estrutura Principal – código não implementado em funções
  - Funções:
    - usage()
    - compare()

- **Ficheiros de teste**
  - Facultado pelo professor:
    - test_a1
    - test_a1.sh
    - test_a1.out
  - Ficheiros utilizados por nós
    - dirold
    - dirnew

Gestão de armazenamento

Funcionalidades

Spacecheck.sh

```
9
10  usage() {
11      echo "Usage: $0 [-n <'.*fileFormat'>] [-d <'Month Day hh:mm'>] [-s <'size'>] [-l <'number of lines'>] [-r] [-a] <dir>
12      -n Sort by file format
13      -d Sort by date
14      -s Sort by size
15      -l Limit of printed lines
16      -r Sort by size, smallest to largest
17      -a Sort by name, alphabetically" #help
18      exit 1
19  }
20
```

Figura 1 – Funcionalidades da função usage()

- **- n | Organiza os ficheiros pelo seu formato**
- **-d | Organiza os ficheiros por data**
- **-s | Organiza os ficheiros por tamanho**
- **-l | Define o número de ficheiros apresentados**
- **-r | Organiza os ficheiros por tamanho crescente**
- **-a | Organiza os ficheiros por nome, alfabeticamente**

Gestão de armazenamento

Funcionalidades

Spacerate.sh

```
function usage() {
    echo "Usage: $0 [-r] [-a] <spacecheck_file1> <spacecheck_file2>
          -r Sort by size, smallest to largest
          -a Sort by name, alphabetically"
    exit 1
}
```

Figura 2 – Funcionalidades da função usage()

- **-r | Organizar por tamanho crescente**

- **-a | Organizar por nome, alfabeticamente**

Gestão de armazenamento

Instruções de uso

Spacecheck.sh

- **-n** | Para listar os ficheiros no diretório escolhido com o formato ".txt" por ordem decrescente do seu tamanho

```
./spacecheck.sh -n ".*\.txt" /caminho/para/seu/diretorio
```

- **-d** | Para listar os ficheiros no diretório escolhido, por data, tendo esta de ser introduzida num formato semelhante ao representado na imagem abaixo

```
./spacecheck.sh -d "Nov 10 10:00" /caminho/para/seu/diretorio
```

- **-s** | Para listar os ficheiros no diretório escolhido com tamanho superior ao inserido, por ordem decrescente

```
./spacecheck.sh -s 1000 /caminho/para/seu/diretorio
```

- **-l** | Para limitar o número de diretórios que são impressos no terminal

```
./spacecheck.sh -l 3 /caminho/para/seu/diretorio
```

Gestão de armazenamento

Instruções de uso

- **-r** | Para listar os ficheiros no diretório escolhido, por ordem crescente

```
./spacecheck.sh -r /caminho/para/seu/diretorio
```

-a| Para listar os ficheiros do diretório escolhido, por ordem alfabética

```
./spacecheck.sh -a /caminho/para/seu/diretorio
```

Spacerate.sh

- **-r** | Para comparar dois ficheiros e listar as suas diferenças, por ordem crescente

```
./spacerate.sh -r file1 file2
```

- **-a** | Para comparar dois ficheiros e listar as suas diferenças, por ordem alfabética

- **Nota:** Também é possível comparar os dois ficheiros sem introduzir qualquer critério de comparação. Esta opção pode ser traduzida da seguinte forma:

```
./spacerate.sh file1 file2
```

Gestão de armazenamento

Restrições de uso

Formato dos ficheiros a comparar (*spacerate.sh*)

O *script* espera que os ficheiros a comparar sigam um formato específico. Os ficheiros devem incluir duas colunas separadas por *tabs* (tamanho do diretório e o *path*). Qualquer desvio deste padrão pode levar a erros no processamento

Uso de "NA" para tamanhos não disponíveis

Por vezes não é possível obter o tamanho dos diretórios. Nestes casos, o *spacecheck.sh* tem ordens para atribuir o valor "NA" ao tamanho do diretório

Nomes de diretórios únicos:

Os nomes dos diretórios devem ser exclusivos e não podem ser duplicados nos ficheiros que salvaguardam as saídas do comando *spacecheck.sh* . Caso contrário, as diferenças podem não ser calculadas corretamente.

Existência dos argumentos para comparação

Os registos das saídas geradas pelas execuções anteriores do *spacecheck.sh* devem existir no sistema e serem fornecidos como argumentos para o *spacerate.sh*. Caso um ou ambos os arquivos estejam ausentes, a comparação não será possível.

Restrições de uso

Estrutura de Diretórios Mantida

O programa presume que a estrutura dos diretórios é mantida entre as execuções do *spacecheck.sh* . Se a estrutura de diretórios mudar entre as execuções, a comparação não refletirá com precisão as mudanças reais de ocupação de espaço.

Processo de validação

Para verificar se os *outputs* de ambos os programas estavam corretos, fomos repetitivamente executando o código, usando como teste o diretório destinado ao projeto (fig. 3).

```
ttabelhaxd@ttabelhaxd:~/SO/projeto1$ ./spacecheck.sh ..
SIZE    NAME    20231110        ..
4125282 ../guioes
7905    ../aula06
3519    ../aula03
2838    ../aula04
1943    ../aula05
1733    ../aula02
0       ../aula02/d
NA      ..
NA      ../projeto1
NA      ../projeto1/semPerms
```

Figura 3 – Exemplo do diretório usado

Gestão de armazenamento

Todas as funcionalidades foram testadas de acordo com as condições mencionadas nas Instruções de Uso.

Para nos certificarmos que estava tudo de acordo com o pedido no guião, relativamente ao *spacerate.sh,* foi necessário redirecionar os *outputs* da execução do *spacecheck.sh* com os argumentos passados, tal como está abaixo representado.

```
ttabelhaxd@ttabelhaxd:~/SO/projeto1$ ./spacecheck.sh /caminho/para/seu/diretorio/dir1 >dir_now
```

Deste passo resulta um ficheiro que poderá agora ser usado para o teste da segunda parte do guião, como exemplificado nas figuras 4 e 5.

```
ttabelhaxd@ttabelhaxd:~/SO/projeto1$ ./spacerate.sh dir_old dir_now
SIZE    NAME
2838    ../aula04 NEW
1943    ../aula05 NEW
1733    ../aula02 NEW
0       ../aula02/d NEW
0       ../aula03
0       ../aula06
0       ../guioes
NA      ..
NA      ../projeto1
NA      ../projeto1/test NEW
NA      ../teste2 REMOVED
-1134   ../aula07 REMOVED
-13423  ../aula08 REMOVED
```

```
ttabelhaxd@ttabelhaxd:~/SO/projeto1$ ./spacerate.sh dir_now dir_old
SIZE    NAME
13423   ../aula08 NEW
1134    ../aula07 NEW
0       ../aula02/d REMOVED
0       ../aula03
0       ../aula06
0       ../guioes
NA      ..
NA      ../projeto1
NA      ../projeto1/test REMOVED
NA      ../teste2 NEW
-1733   ../aula02 REMOVED
-1943   ../aula05 REMOVED
-2838   ../aula04 REMOVED
```

Fig. 4 e 5 – Diferenças entre diretórios

Processo de validação

Como pode acima ser verificado, tivemos o cuidado de garantir que não existia qualquer diferença na ordem em que eram inseridos os dois conteúdos dos diretórios.

Assim, evita-se a necessidade de criar uma restrição para o utilizador no que toca à introdução dos dados a comparar, deixando essa decisão ao arbítrio do utilizador.

É também possível verificar que alguns tamanhos aparecem referenciados como "NA". De modo a que fosse possível simularmos estes casos, foi necessária a remoção da permissão para a execução de um diretório através do seguinte comando:

```
chmod -x diretório
```

Ação que pode ser revertida, substituindo o "-" que antecede o "x" por um "+"

Gestão de armazenamento

Desafios e Soluções

Inexistência do "NA"

Inicialmente, na função *calc_dir_space()* tínhamos definido que todo o valor diferente de 0 deveria ser tomado como "NA", no entanto o resultado obtido não ia de acordo com o pretendido.

```
calc_dir_space() {
    local dir="$1"
    local total_space=0

    while read file; do
        file_space=$(calc_file_space "$file")
        if [ $? -ne 0 ]; then
            total_space="NA"
            break;
        else
            total_space=$((total_space + file_space))
        fi
    done < <(find "$dir" -type f -regex "$regex") #encontra todos os ficheiros que correspondem ao regex

    echo "$total_space"

}
```

Fig.6 – Função calc_dir_space()

Para dar a volta à situação, foi necessária a implementação de uma outra função, *calc_file_space(),* destinada simplesmente à obtenção do tamanho do ficheiro em *bytes*, sendo o seu *return*, o valor a ser comparado na primeira função composta.

```
calc_file_space() {
    du -b "$1" 2>/dev/null | cut -f1  #calcula o tamanho do ficheiro em bytes
    return "${PIPESTATUS[0]}"
}
```

Fig. 7 – Função calc_file_space()

Gestão de armazenamento

Desafios e Soluções

Obtenção ineficiente da data

Numa primeira fase de obtenção da data de um arquivo, recorremos a uma funcionalidade, ***awk***.

No entanto, o ***awk*** é usado para executar ações específicas em linhas de texto, com base em padrões e critérios definidos pelo utilizador, o que não ia de encontro ao pretendido.

Após alguma pesquisa, decidimos que a melhor opção seria usar a função ***-newermt***, derivada do comando ***find*** que acabou por ser exatamente a peça que completou o processo de obtenção da data (Fig. 8).

```
for dir in "$@"; do
    if [ -d "$dir" ]; then
        if [ -z "$date" ]; then #se a data não for especificada
            find "$dir" -type d | while read line; do
            echo -e "$(calc_dir_space "$line")\t$line\t"
        done | sort $sort | awk -v size="$size" '{if ($1 >= size) print $1"\t"$2"\t"}' | head -$limit_lines
        else
            find "$dir" -type d -newermt "$date" | while read line; do
            echo -e "$(calc_dir_space "$line")\t$line\t"
        done | sort $sort | awk -v size="$size" '{if ($1 >= size) print $1"\t"$2"\t"}' | head -$limit_lines
        fi
    else
        echo "$dir: O diretório não existe."
        usage
    fi
```

Fig. 8 – Excerto de código que implementa a função -newermt

Acabou por tornar o código mais eficiente, simples e, consecutivamente, mais facilmente percetível.

Gestão de armazenamento

Desafios e Soluções

Armazenamento dos *paths* e *sizes* (*spacerate.sh*)

Por último, a grande adversidade que tivemos foi arranjar uma maneira prática, porém válida, para guardar os *paths* e os *sizes*.

Numa primeira abordagem, optámos por usar listas, o que acabou por se revelar uma escolha totalmente errada e que por consequência atrasou o projeto, na medida em que tentámos insistir nesta alternativa ainda que não fosse a mais adequada.

Após verificarmos que o código estava demasiado extenso, tivemos de procurar uma alternativa mais curta e simples, cujo papel foi assumido pelos dicionários (fig.9) .
A dualidade *key-value* adequou-se na perfeição ao exemplo que tínhamos em causa e decidimos então que esta seria o caminho a seguir.

Gestão de armazenamento

Desafios e Soluções

```
for i in "${!file1_lines[@]}"; do #para os diretorios que existem nos dois ficheiros
    for j in "${!file2_lines[@]}"; do
        if [[ "$i" == "$j" ]]; then
            if [[ ${file1_lines[$i]} == "NA" ]] || [[ ${file2_lines[$i]} == "NA" ]]; then #se um dos diretorios tiver tamanho NA o re
                echo -e "NA\t$i"
            else
                echo -e "$((${file1_lines[$i]}-${file2_lines[$i]}))\t$i"
            fi
            break
        fi
    done
done

for i in "${!file2_lines[@]}"; do #para os diretorios que existem no file2 mas não no file1 REMOVED
    if [[ ! "${file1_lines[$i]}" ]]; then
        if [[ ${file2_lines[$i]} == "NA" ]]; then
            echo -e "NA\t${i} REMOVED"
        else
            diff=$((file1_lines[$i] - file2_lines[$i]))
            echo -e "${diff}\t${i} REMOVED"
        fi
    fi
done

for i in "${!file1_lines[@]}"; do #para os diretorios que existem no file1 mas não no file2 NEW
    if [[ ! "${file2_lines[$i]}" ]]; then
        if [[ ${file1_lines[$i]} == "NA" ]]; then
            echo -e "NA\t${i} NEW"
        else
            echo -e "${file1_lines[$i]}\t${i} NEW"
        fi
    fi
done
```

Fig. 9 – Excerto de código da implementação do dicionário utilizado

Gestão de armazenamento

Conclusão

Encerrado o projeto, podemos concluir que foram postas à prova as nossas capacidades no que toca à programação em *bash*, mas acima de tudo, as competências de trabalho em grupo, entreajuda e superação de adversidades no decorrer do projeto.

O trabalho teve um impacto substancial no que toca à aquisição de conhecimento nesta linguagem de programação, pois obrigou-nos a "desmontar" o código, de modo a compreendermos o que realmente está a acontecer, o que implica estarmos a par e bem informados nestes tópicos.

De uma forma muito resumida, podemos afirmar que o trabalho foi muito positivo em todos os aspetos.

Gestão de armazenamento

Referências

- https://pt.m.wikipedia.org
- https://unix.stackexchange.com/
- https://www.geeksforgeeks.org