deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

Gestão de armazenamento: Monitorização do espaço ocupado

Desenvolvimento de Scripts em Bash: Spacecheck.sh e Spacerate.sh

DETI - Departamento de Eletrónica, Telecomunicações e Informática

Trabalho realizado por:

Gabriel Martins Silva (113786)
Filipe Ramalho de Oliveira (114640)

Ano Letivo: [2023/2024]

universidade de aveiro

Índice

Introdução

O nosso trabalho trata-se do desenvolvimento de scripts em bash com o objetivo de arranjar uma solução que nos permite monitorizar o espaço ocupado em disco e a sua variação ao longo do tempo, permitindo ao utilizador visualizar o total do espaço ocupado em disco por todos os ficheiros selecionados nas respetivas diretorias passadas como parâmetro ao executar o script.

Fizemos uso da linguagem de script Bash, essencial em sistemas Unix e Linux. O Bash, conhecido pela sua eficácia em automação e manipulação de arquivos, é uma ferramenta versátil para resolver variados desafios computacionais.

Para o desenvolvimento do nosso trabalho criamos dois scripts: spacecheck.sh e spacerate.sh. O script spacecheck.sh é a base deste projeto, e permite ao utilizador ver e monitorizar o espaço ocupado por ficheiros dentro de uma diretoria.

O script spacerate.sh complementa o script spacecheck.sh e fornece ao utilizador uma análise comparativa em relação à evolução do espaço ocupado em disco.

Em conjunto a utilização destes dois scripts permitem ao utilizador uma simplificação da gestão de armazenamento.

O nosso relatório irá mostrar como foi feito o desenvolvimento destes dois scripts tal como as metodologias utilizadas, que aplicámos para resolver o problema.

Desenvolvimento

2) Spacecheck.sh: Descrição e Uso

2.1) Argumentos de entrada:

Este script começa com a verificação da existência de argumentos passados para o programa, quando este é executado, de forma que, se não existirem argumentos, o script imprime uma mensagem de erro. A variável "dir" irá armazenar o último argumento da função, que representa o diretório que programa irá analisar, "opts" irá armazenar os argumentos que definem o que o script irá fazer.

```
# Verifica se existem argumentos
if [ $# -eq 0 ]; then
    echo "Arguments required."
    exit 1
fi

dir="${@: -1}" #ultima palavra
opts=$*
```

Fig. 1 - Verificação da existência de argumentos

Existem várias opções que manipulam o programa para obter diferentes tipos de resultados:

Opção '-n (regex)'

Esta opção é utilizada para quando o utilizador quer encontrar diretórios que contenham um ficheiro de um certo tipo. Se o programa verifica que esta opção é utilizada, através do uso do comando "sed", vai extrair o padrão fornecido após a opção. O padrão é passado para o comando "find", através do uso da opção "-regex", que vai filtrar os diretórios que contenham ficheiros do padrão especificado.

```
# Encontra diretórios com ficheiros com um padrão de nome específico
if [[ "$opts" == *"-n "* ]]; then
  regex_pattern=$(echo "$opts" | sed -n 's/.*-n \([^ ]*\).*/\1/p')
  name_opt="-regex $regex_pattern"
fi
```

Fig. 2 - Leitura da opção -n

Opção ‘-d (data máxima)’

Esta opção é utilizada para quando o utilizador quer encontrar ficheiros que tenham sido modificados até uma certa data. Se o programa verifica que esta opção é utilizada, vai extrair o argumento da data através de um ciclo “while” e do comando “case”. Após a extração do argumento, o programa transforma a variável num padrão “YYYY-MM-DD HH:MM” com recurso ao comando date. O padrão é passado para o comando “find”, que com o uso das opções “-not -newermt", vai filtrar apenas os diretórios modificados até essa data.

```
if [[ "$opts" == *"-d "* ]]; then
  # Extrai argumento de data
  while [[ "$#" -gt 0 ]]; do
    case $1 in
      -d) date_argument="$2"; shift ;;
    esac
    shift
  done

  # Converte o formato "Sep 10 10:00" para "YYYY-MM-DD HH:MM"
  converted_date=$(date -d "$date_argument" "+%Y-%m-%d %H:%M")
  if [[ "$converted_date" == "" ]]; then
    echo "Data inválida"
    exit 1
  fi
fi
```

Fig. 3 - Leitura da opção -d e conversão da data

Opção ‘-s (nº mínimo de bytes)’

Esta opção é utilizada para quando o utilizador quer encontrar diretórios com ficheiros que tenham um tamanho acima de um tamanho especificado. Se o programa verifica que esta opção é utilizada, através do uso do comando “sed”, isola o valor numérico e este é passado para o “find”.

```
if [[ "$opts" == *"-s "* ]]; then
  size_min=$(echo "$opts" | sed -n 's/.*-s \([^ ]*\).*/\1/p')
  size_opt="-size +${size_min}c" # define a opção de tamanho mínimo
fi
```

Fig. 4 - Leitura da opção -s

2.2) Processamento e filtragem de diretórios:

Após colecionar os argumentos e a configuração das opções de filtragem, o script procede para a identificação dos diretórios a serem analisados. Utilizando o comando “find”, o script executa uma busca no diretório alvo, listando todas as suas diretórias. Esta lista é filtrada para que só apareça o que interessa ao utilizador e garante que cada diretório seja processado uma única vez (sort -u).

Caso a busca resulte numa lista vazia, o script conclui que não existem arquivos ou diretórios que correspondam aos critérios fornecidos e notifica o usuário, encerrando a execução com uma mensagem de erro. Se forem encontrados diretórios relevantes, o script segue para a fase de cálculo do espaço em disco.

```
# Encontra todos os diretórios
mapfile -t directories < <(find "$dir" -type d 2>/dev/null | sort -u)

if [[ ${#directories[@]} -eq 0 ]]; then
  echo "No directories found"
  exit 1
else
```

Fig. 5 - Procura de diretórios e verificação

2.3) Cálculo do espaço ocupado no disco:

Para cada diretório identificado, o script inicializa a variável “sum” como zero, que servirá para acumular o tamanho total dos arquivos que satisfaçam os critérios de seleção. Se a opção ‘-d’ foi ativada, o script aplica um filtro adicional para incluir apenas arquivos modificados até a data especificada. De forma similar, a opção ‘-n’ aplica um filtro de nome de arquivo, e a opção ‘-s’ filtra pelo tamanho mínimo do arquivo.

```
mapfile -t directories_size < <(
  for d in "${directories[@]}"; do
    sum=0
    if [[ "$opts" == *"-d "* ]]; then
      # Usa a data convertida com find -not -newermt para filtrar ficheiros mais antigos
      mapfile -t filtered_list < <(
        find "$d" -type f ${name_opt:+$name_opt} -not -newermt "$converted_date" ${size_opt:+$size_opt}
          2>/dev/null | sort -u)
    else
      mapfile -t filtered_list < <(find "$d" -type f ${name_opt:+$name_opt} ${size_opt:+$size_opt}
          2>/dev/null | sort -u)
    fi
```

Fig. 6 - Cálculo do espaço ocupado por diretório

Em seguida, é feita uma nova busca com o comando “find” para listar os arquivos que passam pelos filtros estabelecidos. Para cada arquivo encontrado, o script verifica se tem permissões de leitura; caso contrário, a variável ‘sum’ é marcada como "NA" e a iteração é interrompida.

Se o arquivo é acessível, o script utiliza o comando “du” para obter o tamanho em bytes e adiciona esse valor ao acumulado ‘sum’.

```
  if [[ ${#filtered_list[@]} -gt 0 ]]; then
  # Se existirem ficheiros que passam no filtro
    for e in "${filtered_list[@]}"; do
      if ! [[ -r $e ]]; then # caso nao tenha permissao de leitura
        sum="NA" # nao aplicável
        break
      fi
      space=$(du -s -b "$e" | cut -f1)
      sum=$((sum + space)) # soma o tamanho dos ficheiros que passam no filtro
    done
  fi
  echo "$sum $d" # imprime "(soma) (diretório)"

done | sort -u # remove duplicados
```

Fig. 7 - Cálculo do espaço ocupado por diretório

Finalmente, o script imprime o tamanho acumulado juntamente com o caminho do diretório, proporcionando uma visão clara do espaço utilizado.

2.4) Ordenação de resultados:

Os resultados colecionados são então submetidos ao processo de ordenação conforme definido pelas opções ‘-r’, ‘-a’, ou ‘-ra’ ainda não abordadas neste relatório, produzindo uma saída adaptada às necessidades do utilizador. O script faz uso de “sort” e “awk” para organizar os dados para dentro de um array para este poder ser usado pela função print_array().

```
# Ordena e armazena o output num array
if [[ ${#directories_size[@]} -gt 0 ]]; then
  if [[ "$opts" == *"-a "* ]] || [[ "$opts" == *"-ra "* ]]; then
    mapfile -t array < <(printf "%s\n" "${directories_size[@]}" | awk -F'/' '{print NF-1, $0}' | eval "$sort_cmd" | cut -d' ' -f2-)
  else
    mapfile -t array < <(printf "%s\n" "${directories_size[@]}" | eval "$sort_cmd")
  fi
else
  echo "No files found"
  exit 1
fi
```

Fig. 8 – Ordenação dos diretórios

2.5) Impressão dos dados:

O script termina com a função print_array(), que é responsável por apresentar os dados finais ao usuário de forma organizada. Esta função imprime o cabeçalho e os dados do espaço em disco de cada diretório, respeitando o limite de linhas caso a opção '-l' tenha sido usada pelo utilizador.

Utilizando o array criado no processo de ordenação de resultados e utilizando-o como argumento na função print_array(), o programa satisfaz todas as necessidades do utilizador.

```
print_array array # imprime o array
```

Fig. 9 – Chamada de print_array()

3) Spacerate.sh: Descrição e uso

3.1) Argumentos de entrada:

O script 'spacerate.sh' foi concebido para comparar o uso de espaço em disco ao longo do tempo, tomando como base dados gerados pelo 'spacecheck.sh'. No início da execução, o 'spacerate.sh' garante que foram fornecidos dois argumentos - os ficheiros que representam os dados de dois momentos distintos.

```
# Verifica se existem pelo menos 2 argumentos
if [ $# -lt 2 ]; then
    echo "Missing at least two arguments."
    exit 1
fi
```

Fig. 10 – Verificação de argumentos mínimos

As variáveis 'filenew' e 'fileold' são atribuídas aos dois últimos argumentos passados, que correspondem aos ficheiros de entrada mais recente e mais antigo, respetivamente. A variável 'opts' continua a armazenar todas as opções de linha de comando que influenciam o comportamento do script.

```
filenew=${@: -2:1} #penultima palavra
fileold=${@: -1} #ultima palavra
opts=$*
```

Fig. 11 – Uso dos argumentos

3.2) Comparação de dados:

O script começa por verificar a existência dos ficheiros especificados. Se ambos existirem, ele lê e armazena as linhas de cada ficheiro em arrays separados, excluindo a primeira linha, que é a linha do cabeçalho.

```
#ler os ficheiros
if [ -f "$fileold" ] && [ -f "$filenew" ]; then
# Verifica se os ficheiros existem
  mapfile -t linesold < <(
    firstLine=true
    # Ler o ficheiro linha por linha
    while IFS= read -r line
    do
      if [ "$firstLine" = true ]; then
        firstLine=false
        continue # Ignora a primeira linha
        fi
        echo "$line"
    done < "$fileold"
  )
  mapfile -t linesnew < <(
    firstLine=true
    # Ler o ficheiro linha por linha
    while IFS= read -r line
    do
      if [ "$firstLine" = true ]; then
        firstLine=false
        continue # Ignora a primeira linha
        fi
        echo "$line"
    done < "$filenew"
  )
else
  echo "File not found"
  exit 1
fi
```

Fig. 12 – Leitura dos ficheiros

Com os dados armazenados, o script executa um loop que compara as linhas dos dois arrays. Para cada linha do ficheiro antigo, procura-se uma correspondência no ficheiro mais recente.

Quando uma correspondência é encontrada, calcula-se a diferença de espaço utilizado e registra-se a informação. Se não for encontrada uma correspondência, isso indica que o arquivo foi removido e a sua diferença de tamanho é marcada como tal.

```
mapfile -t array < <(
  for (( i = 0; i < ${#linesold[@]}; i++ )); do
    found=false
    for (( j = 0; j < ${#linesnew[@]}; j++ )); do
      # se o nome do ficheiro for igual
      if [ "$(echo "${linesold[i]}" | cut -d ' ' -f 2-)" = "$(echo "${linesnew[j]}" | cut -d ' ' -f 2-)" ]; then
        found=true
        sizeold=$(echo "${linesold[i]}" | cut -d ' ' -f 1)
        sizenew=$(echo "${linesnew[j]}" | cut -d ' ' -f 1)
        unset 'linesnew[j]' #remove a linha do array para não ser comparada novamente
        break
      fi
    done
```

Fig. 13 – Compara as linhas

Após processar todas as linhas do ficheiro antigo, o script verifica se há novas linhas no ficheiro mais recente, ou seja, diretórios que não existiam anteriormente, marcando-os como "NEW", tal como verifica se no ficheiro antigo há linhas que não existem no ficheiro mais recente, marcando esses diretórios como “REMOVED”

```
    if [ "$found" = "true" ]; then #se a linha foi encontrada
      echo "$((sizeold - sizenew)) $(echo "${linesold[i]}" | cut -d ' ' -f 2-)"
    elif [ "$found" = "false" ]; then #se a linha não foi encontrada (foi removida)
      sizeold=$(echo "${linesold[i]}" | cut -d ' ' -f 1)
      echo "$((0 - sizeold)) $(echo "${linesold[i]}" | cut -d ' ' -f 2-) REMOVED"
    fi
  done

  #verificar se existem ficheiros novos
  for (( i = 0; i < ${#linesnew[@]}; i++ )); do
    if [ -n "${linesnew[i]}" ]; then
      sizenew=$(echo "${linesnew[i]}" | cut -d ' ' -f 1)
      echo "$sizenew $(echo "${linesnew[i]}" | cut -d ' ' -f 2-) NEW"
    fi
  done
)
```

Fig. 14 – Cálculo do espaço e construção das linhas

3.3) Ordenação de resultados:

Tal como no ‘spacecheck.sh’, as opções de ordenação ‘-r’, ‘-a’, e ‘-ra’ são usadas para determinar a ordem em que os resultados serão apresentados. Iremos falar sobre estas opções posteriormente.

3.4) Impressão dos dados:

Tal como no script anterior função print_array() é chamada no final para imprimir os resultados processados. Esta função verifica novamente se a opção ‘-l’ foi usada para limitar o número de linhas da saída. Os resultados da comparação, arquivos que cresceram, diminuíram ou foram removidos/adicionados são apresentados ao utilizador.

4) Elementos comuns aos dois scripts

4.1) Função print_array():

A função print_array() é responsável pela impressão formatada dos dados.

Esta função começa por verificar se a opção '-l' é utilizada. Caso esta opção for utilizada, a função delimita o número de linhas (variável nlines) para o número utilizado depois da opção. Se a opção '-l' não for utilizada o número de linhas (nlines) toma o valor do comprimento total do array.

A função usa um for loop para imprimir os elementos do array (arr), e utiliza 'xargs' para remover os espaços em branco em cada linha de modo que a impressão do array (arr) seja mais legível.

```
print_array() {
  local nlines
  local -n arr=$1  # Usa uma referência de nome para referir a matriz

  if [[ "$opts" == *"-l "* ]]; then
    nlines=$(echo "$opts" | sed -n 's/.*-l \([^ ]*\).*/\1/p')
  else
    nlines=${#arr[@]}
  fi
  for (( i = 0; i < nlines && i < ${#arr[@]}; i++ )); do
    trimmed_element=$(echo "${arr[i]}" | xargs)  # Remove whitespaces no início e no final
    echo "$trimmed_element"
  done
}
```

Fig. 15 – Função print_array()

4.2) Opções sort:

O tipo de “sort” depende do que é passado como argumento de entrada para o programa. Por defeito o espaço ocupado pelos diretórios é organizado de forma decrescente (sort_cmd = “sort -nr”), mas, dependendo das opções, o programa tem diferentes maneiras de dar “sort”:

Opção ‘-r’ (sort_cmd = “sort -n”) → Faz com que o resultado seja organizado por ordem reversa ao “sort” padrão, ou seja, o programa lista os espaços ocupados pelos diretórios de forma crescente.

Opção ‘-a’ (sort_cmd = “sort -t ‘/’ -k2,2”)→ Faz com que o resultado seja organizado por ordem alfabética, ou seja, o programa lista os espaços ocupados pelos diretórios por ordem alfabética do nome destes.

Opção ‘-ra’ (sort_cmd = “sort -r -t ‘/’ -k2,2”)→ É uma espécie de junção das duas opções anteriores. Faz com que o resultado seja organizado por ordem reversa à ordem alfabética.

```
# sort options
sort_cmd="sort -nr"

if [[ "$opts" == *"-r "* ]]; then
  sort_cmd="sort -n"

elif [[ "$opts" == *"-a "* ]]; then
  sort_cmd="sort -t '/' -k2,2"

elif [[ "$opts" == *"-ra "* ]]; then
  sort_cmd="sort -r -t '/' -k2,2"
fi
```

Fig. 16 – Opções de sort

Testes e Validação

Spacecheck.sh

```
gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh test_a1
SIZE NAME 20231113 test_a1
66508 test_a1
40368 test_a1/aaa
9544 test_a1/rrr
7240 test_a1/aaa/zzzz
```

Verificações

```
gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh
Arguments required.

gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh diretório_não_existente
No directories found
```

Opções “sort”

```
gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh -a test_a1
SIZE NAME 20231113 -a test_a1
66508 test_a1
40368 test_a1/aaa
7240 test_a1/aaa/zzzz
9544 test_a1/rrr
```

```
gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh -r test_a1
SIZE NAME 20231113 -r test_a1
7240 test_a1/aaa/zzzz
9544 test_a1/rrr
40368 test_a1/aaa
66508 test_a1

gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh -ra test_a1
SIZE NAME 20231113 -ra test_a1
9544 test_a1/rrr
7240 test_a1/aaa/zzzz
40368 test_a1/aaa
66508 test_a1
```

Opções de filtragem

```
gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh -n ".*txt" test_a1
SIZE NAME 20231113 -n .*txt test_a1
20 test_a1
10 test_a1/rrr
0 test_a1/aaa/zzzz
0 test_a1/aaa
gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh -d "Nov 08 09:25" test_a1
SIZE NAME 20231113 -d Nov 08 09:25 test_a1
198814 test_a1
0 test_a1/rrr
0 test_a1/aaa/zzzz
0 test_a1/aaa

gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacecheck.sh -s 9000 test_a1
SIZE NAME 20231113 -s 9000 test_a1
208348 test_a1
9534 test_a1/rrr
0 test_a1/aaa/zzzz
0 test_a1/aaa
```

Spacerate.sh

```
gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacerate.sh spacecheck_20231106 spacecheck_20231011
SIZE NAME
14695834 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz
14695834 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/
10624580 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs
6425220 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs/Aula03
3530678 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs/Aula04 NEW
1563598 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/sm
1082908 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so
807715 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed
793351 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula02 NEW
606203 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/mpei
297948 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula05 NEW
219192 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/mpei/Guiao02
2093 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula03
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula01
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/sm/Aula04_coefs REMOVED
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/sm/Aula02
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs/Aula05 REMOVED
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs/Aula03/RS_3 REMOVED
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula06 REMOVED
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula01
-26539 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/sm/Aula03 (2) REMOVED
-57946 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula02
-100020 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula06 REMOVED
-146377 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/mpei/Guiao01 REMOVED
-159284 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula05 REMOVED
-213710 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula04 REMOVED
```

```
gabriel-silva@gabriel-silva:~/SO_Projeto$ ./spacerate.sh -a spacecheck_20231106 spacecheck_20231011
SIZE NAME
14695834 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula01
-57946 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula02
-213710 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula04 REMOVED
-159284 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula05 REMOVED
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed/aula06 REMOVED
807715 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/aed
14695834 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz
-146377 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/mpei/Guiao01 REMOVED
219192 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/mpei/Guiao02
606203 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/mpei
6425220 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs/Aula03
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs/Aula03/RS_3 REMOVED
3530678 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs/Aula04 NEW
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs/Aula05 REMOVED
10624580 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/rs
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/sm/Aula02
-26539 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/sm/Aula03 (2) REMOVED
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/sm/Aula04_coefs REMOVED
1563598 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/sm
0 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula01
793351 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula02 NEW
2093 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula03
297948 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula05 NEW
-100020 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so/aula06 REMOVED
1082908 /home/gabriel-silva/Insync/gabrielmsilva4@ua.pt/OneDrive Biz/so
```

Conclusão

Ao longo deste relatório explorámos o desenvolvimento e as funcionalidades dos dois scripts: **spacecheck.sh** e **spacerate.sh.**

Com a realização dos testes pudemos verificar a eficiência e as funcionalidades dos dois scripts. Apesar de ter sido um trabalho onde teve de haver muita autoaprendizagem e dedicação, sentimos que, depois de o realizar, pudemos fortalecer as nossas competências numa linguagem onde não tínhamos muita experiência, contudo não significa que não haverá espaço para possíveis melhorias.