



Sistemas Operativos

2024/25

Trabalho 1

Ferramenta de criação/atualização de cópias de segurança em bash

Tiago Coelho 118745

Bernardo Lázaro 119230

Professores:

Nuno Lau (nunolau@ua.pt)

Guilherme Campos (guilherme.campos@ua.pt)

Índice

Introdução.....	3
0. Indicação do interpretador.....	4
1. Funções.....	5
1.1 checkExistance.sh	5
1.2 checkFile.sh.....	6
1.3 checkPath.sh	6
1.4 isNewer.sh	7
2. Flags.....	8
2.1 <i>Flag</i> -c.....	8
2.2 <i>Flag</i> -b [tfile]	8
2.3 <i>Flag</i> -r [regexp].....	8
3. Código	9
3.1 backup_files.sh	9
Inicialização e sources	9
Tratamento de <i>flags</i>	9
Asserções Iniciais.....	10
Verificação de \$bkp	10
Remoção de ficheiros.....	11
Cópia de ficheiros.....	11
3.2 backup.sh	13
Inicialização e sources	13
Tratamento de <i>flags</i>	13
Asserções Iniciais.....	14
Verificação de \$bkp	15
Remoção de ficheiros.....	15
Cópia de ficheiros.....	16
Fim do programa	17
3.3 backup_summary.sh	18
Inicialização e sources	18
Tratamento de <i>flags</i>	19
Asserções Iniciais.....	19
Verificação de \$bkp	20
Remoção de ficheiros.....	20
Cópia de ficheiros.....	21
Verificações finais.....	23
Impressão do resultado	23
Fim do programa	23
3.4 backup_check.sh	24
Inicialização e sources	24

Asserções Iniciais	24
Entrada recursiva em sub-diretórios.....	25
Comparação de ficheiros	25
Fim do programa.....	25
6. Testes (<i>a priori</i>).....	26
6.1 backup_files.sh	26
SOURCE.....	26
Testes Válidos	26
Testes Inválidos.....	27
6.2 backup.sh.....	28
SOURCE.....	28
exclude_list.txt.....	28
Testes Válidos	29
Testes Inválidos.....	33
6.3 backup_summary.sh	35
SOURCE.....	35
exclude_list.txt.....	35
Testes Válidos	35
Testes Inválidos.....	42
SOURCE.....	45
Pastas iguais	46
Testes Válidos	46
Testes Inválidos.....	46
Pastas diferentes (Ficheiros alterados em BACKUP)	47
Testes (<i>a posteriori</i>).....	48
backup_files.sh	48
backup.sh	49
Edição de ficheiros em SOURCE (edição de “file5.html”, “file9.pdf” e “literally some space”)	49
Adição e remoção de ficheiros em SOURCE (adição de “some new_file.c” e remoção de “just another space.csv”)	49
backup_summary.sh	50
Edição de ficheiros em SOURCE (edição de “file5.html”, “file9.pdf” e “literally some space”)	50
Adição e remoção de ficheiros em SOURCE (adição de “some new_file.c” e remoção de “just another space.csv”)	51
Versões Anteriores	52
Problemas e Abordagens para resolver	52
Bibliografia.....	56
Conclusão	57

Introdução

Este projeto surge no contexto da Unidade Curricular de Sistemas Operativos, da Licenciatura em Engenharia Informática no ano letivo 2024/25.

No enunciado do trabalho, foi-nos proposta a realização de uma ferramenta de criação/atualização de uma cópia de segurança (*backup*) na linguagem *bash*.

Para a realização deste projeto foi utilizado um repositório no GitHub de forma a acompanhar o progresso do trabalho e também de forma a saber o que ainda estava por implementar.

0. Indicação do interpretador

```
1      #!/bin/bash
```

Figura 0 – Indicação do interpretador

Antes de qualquer programa, é-se iniciado o código especificando à *shell* qual o interpretador a ser utilizado, neste caso o **#!/bin/bash**. Esta linha é por vezes conhecida como **shebang**.

1. Funções

1.1 checkExistance.sh

```
1  #!/bin/bash
2
3  function checkExistance() {
4      if [ -e "$1" ]; then
5          return 1
6      else
7          return 0
8      fi
9 }
```

Figura 1 - Código da função *checkExistance()*

Argumento:

- \$1 – path

O propósito da função *checkExistance()* é verificar se um certo ficheiro existe. Se existir, a função retorna 1, caso contrário retorna 0.

No contexto deste projeto esta função será utilizada para verificar se **path/to/bkp/file** existia antes de ser copiado/atualizado.

1.2 checkFile.sh

```
1  #!/bin/bash
2
3  function checkFile() {
4      # Verifica se o ficheiro ou o seu path corresponde a alguma linha do ficheiro
5      while IFS= read -r line; do
6          if [ "$3" == "$line" ] || [ "$2" == "$line" ]; then
7              return 1
8          fi
9      done < "$1"
10     return 0
11 }
```

Figura 2 - Código da função *checkFile()*

Argumentos:

- \$1 – path/to/excluded_list.txt
- \$2 – path/to/file
- \$3 – file_name

O propósito da função *checkFile()* é verificar se um certo ficheiro tem correspondência com algum nome ou caminho para ficheiro incluídos num ficheiro de texto. Se o ficheiro corresponder, então a função retorna 1, caso contrário retorna 0.

No contexto deste projeto será utilizada para verificar se um ficheiro é “proibido” de ser copiado (aquando da utilização da flag *-b*). Isto acontecerá quando o seu nome/caminho esteja contido na lista “**excluded_list.txt**”.

1.3 checkPath.sh

```
1  #!/bin/bash
2
3  # Função que verifica se o path indicado existe
4  function checkPath() {
5      if [ -d "$1" ]; then
6          return 1
7      else
8          return 0
9      fi
10 }
```

Figura 3 - Código da função *checkPath()*

Argumentos:

- \$1 – path

O propósito da função *checkPath()* é verificar se um certo caminho existe como diretório. Se existir, a função retorna 1, caso contrário retorna 0.

No contexto deste projeto será utilizada para verificar se **\$src** e **\$bkp** existem.

1.4 isNewer.sh

```
1  #!/bin/bash
2
3  function isNewer() {
4      # Caso ainda exista, verificamos se é mais recente ou não
5      if [[ "$1" -nt "$2" ]]; then
6          return 1
7      else
8          return 0
9      fi
10 }
```

Figura 4 - Código da função *isNewer()*

Argumentos:

- \$1 – path/to/*file1*
- \$2 – path/to/*file2*

O propósito da função *isNewer()* é verificar se, entre dois ficheiros, um é mais recente do que o outro.

Se *file1* for mais recente que *file2*, então a função retorna 1, caso contrário retorna 0.

No contexto deste projeto será utilizada para verificar se, entre dois ficheiros com o mesmo nome, qual é mais recente, o de **\$src** ou de **\$bkp**.

2. Flags

Utilização:

```
./backup.sh [-c] [-b tfile] [-r regexpr] dir_trabalho dir_backup
```

As *flags* são argumentos opcionais e servem para modificar o comportamento normal do programa.

As *flags* implementadas são: *flag c*, *flag b* e *flag r*.

2.1 Flag -c

A *flag -c* tem como objetivo mostrar ao utilizador, através do terminal, que comandos executaria ao longo do programa sem os realmente os executar. Isto foi obtido através de uma verificação condicional. Se \$c_flag for igual a 1, então a *flag -c* está ativa e o programa apenas executa comando echo para simular a sua execução. Por outro lado, se \$c_flag for igual a 0, então a *flag -c* está inativa pelo que o programa decorre normalmente executando os comandos devidos.

2.2 Flag -b [tfile]

A *flag -b* tem como objetivo ignorar ficheiros e/ou caminhos para ficheiro que estejam escritos num certo ficheiro de texto indicado pelo utilizador. Se \$b_flag for igual a 1, então a *flag -b* está ativa e o programa ignorará qualquer ficheiro que corresponda aos nomes e/ou caminhos para ficheiro escritos no ficheiro de texto. Por outro lado, se \$b_flag for igual a 0, então a *flag -b* está inativa pelo que o programa decorre normalmente sem ignorar qualquer ficheiro.

2.3 Flag -r [regexpr]

A *flag -r* tem como objetivo especificar que ficheiros devem ser copiados através de uma expressão regular. Se \$r_flag for igual a 1, então a *flag -r* está ativa e o programa apenas copiará ficheiro que verifiquem a expressão regular. Por outro lado, se \$r_flag for igual a 0, então a *flag -r* está inativa pelo que o programa decorre normalmente sem ignorar qualquer ficheiro.

3. Código

3.1 backup_files.sh

Inicialização e sources

```
1  #!/bin/bash
2
3  source ./functions/checkPath.sh
4  source ./functions/isNewer.sh
5
6  # $1 - src (Pasta a copiar)
7  # $2 - bkp (Pasta onde colar) [Criar Pasta caso não exista]
8
9  c_flag=0
```

Figura 5 - Inicialização e sources

Optámos por dar `source` às 2 funções utilizadas na implementação de forma a diminuir o número de linhas de código do programa principal.

Também optámos por usar valores inteiros para definir se a *flag* `-c` está presente no comando ou não (como valor *default* temos `$c_flag=0`).

Tratamento de *flags*

```
11 while getopts ":c" flag; do
12   case "$flag" in
13     c)
14       c_flag=1
15     ;;
16     \?)
17       echo -e "Parâmetros incorretos!\nEsperado: -c /path/to/src /path/to/bkp"
18       exit 1
19     ;;
20   esac
21 done
22
23 # Dá shift das flags e "retira-as" dos argumentos
24 shift $((OPTIND - 1))
```

Figura 6 - Utilização de `getopts`

É usada a função `getopts` para processar opções de linha de comando passadas como argumentos.

Neste caso, o script procura apenas pela opção “`-c`”. Se a opção `-c` for passada, a variável `$c_flag` passa a ter valor igual a 1 indicando que a *flag* `-c` está ativa.

Caso seja passado como argumento um parâmetro incorreto, ou não esperado, o script exibe a seguinte mensagem de erro: “Parâmetros incorretos!”. Sendo também passado de seguida uma sugestão de como devem ser fornecidos os argumentos de forma correta antes de o programa terminar com o comando `exit 1`.

Depois de serem analisadas as *flags*, o comando “`shift $((OPTIND - 1))`” ajusta a posição dos argumentos deslocando a lista de argumentos para a esquerda. Fazendo assim com que o primeiro argumento, que neste caso é a *flag* `-c`, caso esta se encontre ativa, seja descartado. Assim, os próximos argumentos ocuparão as posições `$1` e `$2`.

Isto acontece porque quando a função getopts encontra uma opção (-c no nosso caso), ele incrementa automaticamente o OPTIND (**Option Index**) passando a ter valor igual a 2. Logo o *shift* vai deslocar a lista de argumentos uma vez para a esquerda (2-1=1).

Asserções Iniciais

```
26 # Verifica que o programa tem exatamente 2 argumentos depois de processar as flags (path/to/src e path/to/bkp)
27 if [ $# -ne 2 ]; then
28     echo -e "Parâmetros incorretos!\nEsperado: -c /path/to/src /path/to/bkp"
29     exit 1
30 fi
31
32 # Atribuir os restantes argumentos a src e bkp
33 src="$1"
34 bkp="$2"
35
36 # Verifica se $src e $bkp são o mesmo diretório
37 if [[ $src == "$bkp" ]]; then
38     echo "Os caminhos \"/path/to/src\" e \"\path/to/bkp\" não podem ser iguais!"
39     exit 1;
40 fi
41
42 # Verificar se $src existe
43 checkPath "$src"
44 if [ $? -ne 1 ]; then
45     echo "O caminho \"$src\" não existe!"
46     exit 1;
47 fi
```

Figura 7 - Asserções Iniciais

De forma a prevenir operações sem utilidade nenhuma (no caso de inputs inválidos, por exemplo) implementámos várias verificações (asserções) no início do programa.

Após as *flags* terem sido removidas dos argumentos, verificamos se o número de argumentos era 2, respetivamente *path/to/src* e *path/to/bkp*. Com essa verificação validada, então prosseguimos para a atribuição de \$src e \$bkp.

De seguida, verificámos se \$src e \$bkp não eram a mesma pasta, uma vez que isso poderia causar erros e não faria sentido estar a fazer backup de uma pasta nessa própria pasta.

Finalmente, através do uso da função *checkPath()*, implementámos uma verificação da existência de \$src. Se \$src não existir, então o programa aborta.

Verificação de \$bkp

```
49 # Verificar se $bkp existe
50 checkPath "$bkp"
51 if [ $? -ne 1 ]; then
52     # criar a diretoria
53     if [ ${c_flag} -eq 1 ]; then
54         echo "mkdir -p -v \"$bkp\""
55     else
56         mkdir -p -v "$bkp"
57     fi
58 fi
```

Figura 8 - Verificação da existência de \$bkp

Através do uso da função *checkPath()*, implementámos uma verificação da existência de \$bkp. Se \$bkp não existir, então será criada.

Remoção de ficheiros

```
60 shopt -s nullglob
61 shopt -s dotglob
62 for path in "$bkp"/*; do
63     name=$(basename "$path")      # remove o prefixo do path e deixa apenas o nome do ficheiro
64
65     # Se o ficheiro já não existir em src, então apagamos de bkp
66     if [ ! -e "$src/$name" ]; then
67         if [ ${c_flag} -eq 1 ]; then
68             echo "rm -v \"$path\""
69         else
70             rm -v "$path"
71         fi
72     fi
73 done
```

Figura 9 - Remoção de ficheiros de \$bkp já não existentes em \$src

O comando *shopt* configura opções do Shell que alteram o seu comportamento padrão.

“-s *nullglob*” faz com que bash substitua um padrão de *globbing* vazio, como “\$bkp/*”, por exemplo, por uma lista vazia, ao invés de retornar o próprio padrão, evitando assim erros caso o diretório se encontre vazio.

“-s *dotglob*” faz com que o *globbing* (*) também inclua arquivos e diretórios que começem com ponto (.)

Se certos ficheiros ou diretórios já não se encontrarem em \$src, estes serão removidos de \$bkp.

O ciclo *for* percorre cada ficheiro, presente em \$bkp, passando o caminho completo de cada item à variável \$path que, logo de seguida, com o comando *basename*, é fornecido apenas o nome do ficheiro à variável \$name. Graças a esta extração do nome o processo de comparação com o diretório de origem (\$src) é facilitado.

Com o primeiro *if* é verificado de \$src/\$name ainda exista em \$src, se isso não acontecer, então podemos dar início à remoção de \$path (que se encontra em \$bkp). Antes da remoção é feita a verificação se estamos em modo *checking (flag -c)* ou não. Se sim, então o comando a executar será *echo*, caso contrário a remoção é devidamente efetuada.

Cópia de ficheiros

```
84 for file_path in "$src"/*; do
85     file_name=$(basename "$file_path")      # remove o prefixo do path e deixa apenas o nome do ficheiro
86
87     # Verifica se o ficheiro é mais recente em src do que em bkp (cópia se for mais recente ou se ainda não existir em bkp)
88     isNewer "$src/$file_name" "$bkp/$file_name"
89     if [ $? -eq 1 ]; then
90         if [ ${c_flag} -eq 1 ]; then
91             echo "cp -a -v \"$file_path\" \"$bkp\""
92         else
93             cp -a -v "$file_path" "$bkp"
94         fi
95     fi
96 done
97 shopt -u nullglob
98 shopt -u dotglob
99
100 echo "Fim do programa."
```

Figura 10 - Cópia de ficheiros

A parte inicial do *loop* que vai concretizar a cópia de ficheiros é semelhante à parte inicial da remoção de ficheiros, visto anteriormente, onde, neste caso, o ciclo *for* vai percorrer cada ficheiro em \$src, em vez de \$bkp. Usamos também a mesma estratégia armazenando apenas o nome do ficheiro na variável \$file_name.

Com a função *isNewer()*, verificamos se o ficheiro é mais recente em \$src do que em \$bkp para que, caso seja mais recente ou se ainda não existir em \$bkp, seja copiado. Se a função *isNewer()* retornar 1, valor este que será armazenado em \$?, a cópia é executada, caso contrário, por questões de otimização, nada acontece com o ficheiro, uma vez que já está na sua versão mais atualizada.

No final do programa são desativadas as opções “*nullglob*” e “*dotglob*” que foram ativadas anteriormente na remoção de ficheiros/diretórios.

3.2 backup.sh

Inicialização e sources

```
1  #!/bin/bash
2
3  # Verifica se $INITIAL_CALL não está definida
4  # Ao usarmos export fazemos com que qualquer child process (chamada recursiva) consiga aceder ao valor definido na chamada inicial
5  if [ -z "$INITIAL_CALL" ]; then
6      export INITIAL_CALL=1 # Estamos na chamada inicial
7  else
8      export INITIAL_CALL=0 # Estamos na chamada recursiva
9  fi
10
11 source ./functions/checkPath.sh
12 source ./functions/isNewer.sh
13 source ./functions/checkFile.sh
14
15 # $1 - src (Pasta a copiar)
16 # $2 - bkp (Pasta onde colar) [Criar Pasta caso não exista]
17
18 # Inicializar as flags e argumentos
19 c_flag=0
20 b_flag=0
21 r_flag=0
22 flags=""
```

Figura 11 - Inicialização e sources

Como neste *script* iremos fazer chamadas recursivas, fizemos uso da variável \$INITIAL_CALL para identificar se o script está a ser executado pela primeira vez ou se está a ser chamado recursivamente.

Tal como em *backup_files.sh*, importámos as funções que iremos utilizar ao longo do *script* e inicializámos as diferentes *flags* e argumentos.

Tratamento de *flags*

```
24 # Processar as opções com getopt
25 while getopts "cb:r:" flag; do
26     case $flag in
27         c)
28             flags+="-c"
29             c_flag=1
30             ;;
31         b)
32             b_flag=1
33             blockedFiles="$OPTARG"
34             if [[ ! -f "$blockedFiles" ]]; then
35                 echo "[tfile] tem de ser um ficheiro válido!"
36                 exit 1;
37             fi
38             flags+=" -b $blockedFiles"
39             ;;
40         r)
41             r_flag=1
42             regexpr="$OPTARG"
43             flags+=" -r \"$regexpr\""
44             ;;
45         \? )
46             echo -e "Parâmetros incorretos!\nEsperado: -c -b [tfile] -r [regexpr] /path/to/src /path/to/bkp"
47             exit 1
48             ;;
49     esac
50 done
51
52 # Dá shift das flags e "retira-as" dos argumentos
53 shift ${((OPTIND - 1))}
```

Figura 12 - Utilização de getopt

A função *getopt* vai ler e interpretar as *flags* passadas para o *script* na forma de argumento.

A string “cb:r:” define as opções aceites pelo *script*:

- “c”: Uma *flag* simples sem argumento;

- “b:” Uma *flag* com um argumento obrigatório (-b [tfile]);
- “r:” Outra *flag* com um argumento obrigatório (-r [regexpr]).

A variável \$flags vai armazenar todas as *flags* passadas como argumento para que, posteriormente, o processo de fornecer à chamada recursiva as *flags* passadas seja facilitado.

Caso a *flag* -c seja passada, o *script* adiciona “-c” à variável \$flags e atribuir a \$c_flag o valor 1, indicando que esta *flag* se encontra ativa.

\$blockedFiles vai armazenar o argumento da *flag* -b ([tfile]), anteriormente guardado em \$OPTARG pela função *getopts*.

Para terminar, verifica-se se o argumento passado é válido e, caso não seja, o *script* exibe uma mensagem de erro e aborta o programa com *exit 1*. Se o ficheiro for válido, a *flag* -b e o argumento são adicionados à variável \$flags.

Sendo a *flag* -r passada, o *script* atribui a \$r_flag o valor 1, indicando que esta *flag* se encontra ativa. De seguida é armazenado o argumento da *flag* -r ([regexpr]) na variável \$regexpr e são adicionadas à variável \$flags a *flag* -r e o seu argumento.

Caso o *script* receba uma opção inválida, este exibe uma mensagem de erro e aborta o programa com *exit 1*.

No fim, é usado *shift*, tal como no tratamento de *flags* em *backup_files.sh*, para que os argumentos sejam deslocados para a esquerda, fazendo assim com que sejam removidas as *flags* e os seus argumentos, deixando apenas os argumentos do caminho para \$src e para \$bkp para serem processados pelo *script*.

Asserções Iniciais

```

55  # Verifica que o programa tem exatamente 2 argumentos depois de processar as flags (path/to/src e path/to/bkp)
56  if [ $# -ne 2 ]; then
57  |   echo -e "Parâmetros incorretos!\nEsperado: -c -b [tfile] -r [regexpr] /path/to/src /path/to/bkp"
58  |   exit 1
59  fi
60
61 # Atribuir os restantes argumentos a src e bkp
62 src="$1"
63 bkp="$2"
64
65 # Verifica se $src e $bkp são o mesmo diretório
66 if [[ "$src" == "$bkp" ]]; then
67 |   echo "Os caminhos \"/path/to/src\" e \"path/to/bkp\" não podem ser iguais!"
68 |   exit 1;
69 fi
70
71 # Verificar se $src existe
72 checkPath "$src"
73 if [ $? -ne 1 ]; then
74 |   echo "O caminho \"$src\" não existe!"
75 |   exit 1;
76 fi

```

Figura 13 - Asserções Iniciais

Depois do processamento das *flags*, o programa certifica-se que apenas constam 2 argumentos (*path/to/src* e *path/to/bkp*). Se tal não se verificar o programa aborta com *exit 1* e imprime uma mensagem de erro.

São atribuídas às variáveis \$src e \$bkp os respetivos argumentos e verifica-se se estes representam o mesmo diretório. Caso isso aconteça, o *script* exibe uma mensagem de erro e termina, pois, o caminho de \$src nunca poderá ser igual ao caminho de \$bkp, uma vez que isso pode provocar erros.

Por fim verifica-se se o caminho armazenado em \$src existe, se não existir, o programa aborta.

Verificação de \$b kp

```
78  # Verificar se $b kp existe
79  checkPath "$b kp"
80  if [ $? -ne 1 ]; then
81      # criar a diretoria
82      if [ $c_flag -eq 1 ]; then
83          echo "mkdir -p -v \"$b kp\""
84      else
85          mkdir -p -v "$b kp"
86      fi
87  fi
```

Figura 14 - Verificação da existência de \$b kp

Tal como anteriormente, através do uso da função *checkPath()*, implementámos uma verificação da existência de \$b kp. Se \$b kp não existir, então será criada.

Remoção de ficheiros

```
89 shopt -s nullglob
90 shopt -s dotglob
91 for path in "$b kp"/*; do
92     name=$(basename "$path")    # remove o prefixo do path e deixa apenas o nome do ficheiro
93
94     # Se o ficheiro/diretório já não existir em src, então apagamos de bkp
95     if [ ! -e "$src/$name" ]; then
96         # Verificar se é ficheiro ou diretório
97         if [ -f "$path" ]; then
98             if [ $c_flag -eq 1 ]; then
99                 echo "rm -v \"$path\""
100            else
101                rm -v "$path"
102            fi
103        else
104            if [ $c_flag -eq 1 ]; then
105                echo "rm -r -v \"$path\""
106            else
107                rm -r -v "$path"
108            fi
109        fi
110    fi
111 done
```

Figura 15 - Remoção de ficheiros de \$b kp já não existentes em \$src

A remoção de ficheiros de \$b kp que já não se encontram em \$src é feita de forma idêntica à realizada anteriormente em *backup_files.sh*, com a exceção de que, neste programa, é feita uma verificação adicional que consiste em saber se \$path é um caminho para um ficheiro ou para um diretório. Esta verificação é feita pois, dependendo do que queremos remover, temos (ou não) de usar uma *flag* adicional para este comando.

Se \$path se tratar de um caminho para um ficheiro podemos usar o comando *rm* de forma normal, no entanto, se \$path for um caminho para um diretório temos de adicionar a *flag* -r ao comando *rm*, removendo assim todo o seu conteúdo, incluindo sub-diretórios.

Cópia de ficheiros

Criação de sub-diretórios

```
113  for file_path in "$src/*"; do
114      file_name=$(basename "$file_path")      # Remove o prefixo do caminho e deixa apenas o nome do ficheiro/diretório
115
116      if [ -d "$file_path" ]; then
117          # Criar o diretório de destino correspondente, se não existir
118          if [ ! -d "$bkp/$file_name" ]; then
119              if [ $c_flag -eq 1 ]; then
120                  echo "mkdir -p -v '\$bkp/\$file_name'"
121              else
122                  mkdir -p -v "$bkp/$file_name"
123              fi
124          fi
125
126          # Chama o script de backup recursivamente para esse subdiretório
127          if [ $c_flag -eq 1 ]; then
128              echo "eval \"bash \$0\" \$flags \"\$file_path\" \"\$bkp/\$file_name\""
129          fi
130      eval "bash \$0\" \$flags \"\$file_path\" \"\$bkp/\$file_name\""
```

Figura 16 - Criação de sub-diretórios com recursão

Se a variável \$file_path for um caminho para um diretório e este não existir, então é criado o diretório de destino correspondente.

Posteriormente é invocado o *script* de *backup* recursivamente para esse subdiretório com as mesmas *flags* passadas no início do programa.

Cópia de ficheiros

```
132  else
133      # Verifica se o ficheiro é mais recente em src do que em bkp (cópia se for mais recente ou se ainda não existir em bkp)
134      isNewer "$file_path" "$bkp/$file_name"
135      # Se ainda existir e for mais recente, então copiamos
136      if [ $? -eq 1 ]; then
137          if [ ${_c_flag} -eq 1 ]; then
138              if [ ${_b_flag} -eq 1 ]; then
139                  if [ ${_r_flag} -eq 1 ]; then
140                      if [[ "$file_name" == $regeexpr ]]; then
141                          # Mesmo que o ficheiro verifique a expressão regular, se estiver no ficheiro da flag -b, então é ignorado (Flags: -c -b -r)
142                          checkFile "$blockedFiles" "$file_path" "$file_name"
143                          if [ $? -eq 1 ]; then
144                              echo "Arquivo '\$file_name\' será ignorado (flag -b)."
145                          else
146                              echo "cp -a -v '\$file_path \$bkp'"
147                          fi
148                      else
149                          echo "Arquivo '\$file_name\' será ignorado (flag -r)."
150                      fi
151                  else
152                      # Se o ficheiro estiver no ficheiro da flag -b, então é ignorado (Flags: -c -b)
153                      checkFile "$blockedFiles" "$file_path" "$file_name"
154                      if [ $? -eq 1 ]; then
155                          echo "Arquivo '\$file_name\' será ignorado (flag -b)."
156                      else
157                          echo "cp -a -v '\$file_path\" \"$bkp\""
158                      fi
159                  fi
160              else
161                  if [ ${_r_flag} -eq 1 ]; then
162                      # Se o nome do ficheiro verificar a expressão regular, então copiamos (Flags: -c -r)
163                      if [[ "$file_name" == $regeexpr ]]; then
164                          echo "cp -a -v '\$file_path\" \"$bkp\""
165                      else
166                          echo "Arquivo '\$file_name\' será ignorado (flag -r)."
167                      fi
168                  else
169                      # (Flags: -c)
170                      echo "cp -a -v '\$file_path\" \"$bkp\""
171                  fi
172              fi
173          fi
174      fi
175  fi
```

Figura 17 - Cópia de ficheiros com \${_c_flag}=1

Na continuidade do código anterior, caso a variável \$file_path represente um ficheiro no bloco apresentado na figura 17 e 18.

Com recurso à função *isNewer()* é verificado se o ficheiro é mais recente em \$src do que em \$bkp e caso isso se verifique, a função retorna o valor 1, que de seguida dá inicio ao processo da cópia do ficheiro.

No caso da figura 17, a *flag* -c encontra-se ativa tendo apenas como objetivo mostrar ao utilizador, através do terminal, que processos executaria ao longo do programa, sem realmente os executar.

Caso a *flag* -b também se encontre ativa, o programa vai verificar se alguns dos ficheiros que se encontram em \$blockedFiles coincidem com o ficheiro que se quer copiar. Esta verificação é feita através de “*checkFile*” “\$blockedFiles” “\$file_path” “\$file_name””. Se isso acontecer, mesmo que esses ficheiros correspondam a uma certa expressão regular, caso esta seja passada com a *flag* -r, vão ser ignorados e não serão copiados.

Se a *flag* -r estiver ativa e a *flag* -b não, apenas vão ser copiados ficheiros que verifiquem a expressão regular.

Se só se encontrar a *flag* -c ativa o programa vai simular a cópia de todos os ficheiros, sem quaisquer restrições.

```
173
174     else
175         if [ $b_flag -eq 1 ]; then
176             if [ ${file_name} == $regexpr ]; then
177                 # Mesmo que o ficheiro verifique a expressão regular, se estiver no ficheiro da flag -b, então é ignorado (Flags: -b -r)
178                 checkFile "$blockedFiles" "$file_path" "$file_name"
179                 if [ $? -eq 1 ]; then
180                     echo "Arquivo \"$file_name\" será ignorado (flag -b)."
181                 else
182                     cp -a -v "$file_path" "$bkp"
183                 fi
184             else
185                 echo "Arquivo \"$file_name\" será ignorado (flag -r)."
186             fi
187         else
188             # Se o ficheiro estiver no ficheiro da flag -b, então é ignorado (Flags: -b)
189             checkFile "$blockedFiles" "$file_path" "$file_name"
190             if [ $? -eq 1 ]; then
191                 echo "Arquivo \"$file_name\" será ignorado (flag -b)."
192             else
193                 cp -a -v "$file_path" "$bkp"
194             fi
195         fi
196     else
197         if [ $r_flag -eq 1 ]; then
198             # Se o nome do ficheiro verificar a expressão regular, então copiamos (Flags: -r)
199             if [ ${file_name} == $regexpr ]; then
200                 cp -a -v "$file_path" "$bkp"
201             else
202                 echo "Arquivo \"$file_name\" será ignorado (flag -r)."
203             fi
204         else
205             cp -a -v "$file_path" "$bkp"
206         fi
207     fi
208   fi
209 done
210 shopt -u dotglob
211 shopt -u nullglob
```

Figura 18 - Cópia de ficheiros com \$c_flag=0

Na figura 18, a *flag* -c não se encontra ativa, ou seja, o programa vai comportar-se da mesma maneira vista anteriormente (figura 17) mas neste caso, os processos vão realmente acontecer, ao contrário do bloco anterior, onde apenas eram apresentados no terminal.

Fim do programa

```
215  # Exibir "Fim do programa" apenas na primeira execução (não recursiva)
216  if [ $INITIAL_CALL -eq 1 ]; then
217      echo "Fim do programa."
218  fi
```

Figura 19 - Fim do programa

Graças ao valor guardado no início do programa em \$INITIAL_CALL, é possível verificar que se o valor da variável for 1, significa que nos encontramos na chamada inicial do *script*.

Assim, se nos encontrarmos na chamada inicial, é passada no terminal uma mensagem de “Fim do programa”.

Se o script não conseguisse saber se se encontrava na chamada inicial ou recursiva, sempre que o *script*, nas chamadas recursivas, acabasse, era passada a mensagem errada de fim de programa. Daí termos feito uso da variável \$INITIAL_CALL.

3.3 backup_summary.sh

Inicialização e sources

```
1 #!/bin/bash
2
3 # Verifica se $INITIAL_CALL não está definida
4 # Ao usarmos export fazemos com que qualquer child process (chamada recursiva) consiga aceder ao valor definido na chamada inicial
5 if [ -z "$INITIAL_CALL" ]; then
6 |   export INITIAL_CALL=1 # Estamos na chamada inicial
7 else
8 |   export INITIAL_CALL=0 # Estamos na chamada recursiva
9 fi
10
11 source ./functions/checkPath.sh
12 source ./functions/isNewer.sh
13 source ./functions/checkFile.sh
14 source ./functions/checkExistance.sh
15
16 # $1 - src (Pasta a copiar)
17 # $2 - bkp (Pasta onde colar) [Criar Pasta caso não exista]
18
19 # Inicializar as flags e argumentos
20 c_flag=0
21 b_flag=0
22 r_flag=0
23 flags=""
24
25 errors=0
26 warnings=0
27 updated=0
28 copied=0
29 deleted=0
30
31 copied_size=0
32 deleted_size=0
```

Figura 20 - Inicialização e sources

A inicialização deste *script* é muito semelhante à de *backup.sh* uma vez que na íntegra são o mesmo *script*, mas com algumas nuances de diferença.

Este programa tem como *source* as 3 funções anteriores e uma função extra, sendo esta a função *checkExistance.sh*

Para além disso temos também 7 novas variáveis: \$erros, \$warnings, \$updated, \$copied, \$deleted, \$copied_size e \$deleted_size.

- \$erros – Número de erros ocorridos durante a execução do programa.
- \$warnings – Número de avisos ocorridos durante a execução do programa.
- \$updated – Número de ficheiros atualizados (que já existiam em \$bkp mas foram substituídos com uma versão mais recente proveniente de \$src) durante a execução do programa.
- \$copied – Número de ficheiros copiados (que ainda não existiam em \$bkp e que foram copiado de \$src) durante a execução do programa.
- \$deleted – Número de ficheiros apagados durante a execução do programa.
- \$copied_size – Tamanho total, em bytes, dos ficheiros copiados durante a execução do programa.
- \$deleted_size – Tamanho total, em bytes, dos ficheiros apagados durante a execução do programa.

Tratamento de *flags*

```
34 # Processar as opções com getopt
35 while getopt "cb:r:" flag; do
36     case $flag in
37         c)
38             flags+=" -c"
39             c_flag=1
40             ;;
41         b)
42             b_flag=1
43             blockedFiles="$OPTARG"
44             if [[ ! -f "$blockedFiles" ]]; then
45                 echo "[file] tem de ser um ficheiro válido!"
46                 ((errors+=1))
47             fi
48             flags+=" -b $blockedFiles"
49             ;;
50         r)
51             r_flag=1
52             regexpr="$OPTARG"
53             flags+=" -r \"$regexpr\""
54             ;;
55         \?)
56             echo -e "Parâmetros incorretos!\nEsperado: -c -b [file] -r [regexpr] path/to/src /path/to/bkp"
57             ((errors+=1))
58             ;;
59     esac
60 done
61
62 # Dá shift das flags e "retira-as" dos argumentos
63 shift ${OPTIND-1}
```

Figura 21 - Utilização de getopt

O tratamento de *flags* em *backup_summary.sh* é muito semelhante ao tratamento de *flags* em *backup.sh*, pois as *flags* são as mesmas e representam as mesmas funções. A única diferença, é que, se algum dos parâmetros apresentados for incorreto ou inválido, é incrementado o valor de \$errors em uma unidade.

Asserções Iniciais

```
65 # Verifica que o programa tem exatamente 2 argumentos depois de processar as flags (path/to/src e path/to/bkp)
66 if [ $# -ne 2 ]; then
67     echo -e "Parâmetros incorretos!\nEsperado: -c -b [file] -r [regexpr] path/to/src /path/to/bkp"
68     ((errors+=1))
69 fi
70
71 # Atribuir os restantes argumentos a src e bkp
72 src="$1"
73 bkp="$2"
74
75 # Verifica se $src e $bkp são o mesmo diretório
76 if [[ "$src" == "$bkp" ]]; then
77     echo "Os caminhos \"path/to/src\" e \"path/to/bkp\" não podem ser iguais!"
78     ((errors+=1))
79 fi
80
81 # Verificar se $src existe
82 checkPath "$src"
83 if [ $? -ne 1 ]; then
84     echo "O caminho \"$src\" não existe!"
85     ((errors+=1))
86 fi
```

Figura 22 - Asserções Iniciais

As asserções iniciais são realizadas de forma semelhante às que foram implementadas em *backup.sh*, com a implementação de um contador que vai incrementando à medida que ocorram erros no programa.

Verificação de \$b kp

```
88  if [[ errors -eq 0 ]]; then
89      # Verificar se $b kp existe
90      checkPath "$b kp"
91      if [ $? -ne 1 ]; then
92          # criar a diretoria
93          if [ $c_flag -eq 1 ]; then
94              echo "mkdir -p -v \"$b kp\""
95          else
96              mkdir -p -v "$b kp"
97          fi
98      fi
```

Figura 23 - Verificação da existência de \$b kp

Tal como anteriormente, através do uso da função *checkPath()*, implementámos uma verificação da existência de \$b kp. Se \$b kp não existir, então será criada.

Remoção de ficheiros

```
100 shopt -s nullglob
101 shopt -s dotglob
102 for path in "$b kp"/*; do
103     name=$(basename "$path")
104
105     # Se o ficheiro/diretório já não existir em src, então apagamos de b kp
106     if [ ! -e "$src/$name" ]; then
107         ((deleted+=1))
108         deleted_size=$((deleted_size + $(stat -c%s "$path")))
109         # Verificar se é ficheiro ou diretório
110         if [ -f "$path" ]; then
111             if [ $c_flag -eq 1 ]; then
112                 echo "rm -v \"$path\""
113             else
114                 rm -v "$path"
115             fi
116         else
117             if [ $c_flag -eq 1 ]; then
118                 echo "rm -r -v \"$path\""
119             else
120                 rm -r -v "$path"
121             fi
122         fi
123     fi
124 done
```

Figura 24 - Remoção de ficheiros de \$b kp já não existentes em \$src

Também a verificação da existência de \$b kp e a remoção de ficheiros de \$b kp que já não se encontram em \$src são realizadas de forma semelhante em comparação a *backup.sh*. Na remoção de ficheiros é apenas acrescentado uma variável que vai contar o número de ficheiros apagados durante a execução do programa e também uma variável que vai armazenar o tamanho total, em bytes, dos mesmos.

Cópia de ficheiros

Criação de sub-diretórios

```
126     for file_path in "$src"/*; do
127         file_name=${basename "$file_path"}      # Remove o prefixo do caminho e deixa apenas o nome do ficheiro/diretório
128
129         if [ -d "$file_path" ]; then
130             # Criar o diretório de destino correspondente, se não existir
131             if [ ! -d "$bkp/$file_name" ]; then
132                 if [ $c_flag -eq 1 ]; then
133                     echo "mkdir -p -v \"$bkp/$file_name\""
134                 else
135                     mkdir -p -v "$bkp/$file_name"
136                 fi
137             fi
138
139             # Chama o script de backup recursivamente para esse subdiretório
140             if [ $c_flag -eq 1 ]; then
141                 echo "eval \"bash \"$0\" $flags \"$file_path\" \"$bkp/$file_name\"\""
142             fi
143             eval "bash \"$0\" $flags \"$file_path\" \"$bkp/$file_name\""
```

Figura 25 - Criação de sub-diretórios com recursão

A criação de sub-diretórios, com recursão, é realizada usando também os mesmos processos que em *backup.sh*.

Cópia de ficheiros (tendo em conta as flags)

```

145     else
146         isNewer "$file_path" "$bkp/$file_name"
147         # Se ainda existir e for mais recente, então copiamos
148         if [ $? -eq 1 ]; then
149             checkExistence "$bkp/$file_name"    # Utilizado para, no final, sabermos se o ficheiro foi copiado ou atualizado
150             control=$?
151             var=0
152
153             if [ ${c_flag} -eq 1 ]; then
154                 if [ ${sb_flag} -eq 1 ]; then
155                     if [ ${sr_flag} -eq 1 ]; then
156                         if [ "$file_name" =~ $regexpr ]; then
157                             # Mesmo que o ficheiro verifique a expressão regular, se estiver no ficheiro da flag -b, então é ignorado (Flags: -c -b -r)
158                             checkFile "$blockedFiles" "$file_path" "$file_name"
159                             if [ ${?} -eq 1 ]; then
160                                 echo "Arquivo \"\$file_name\" será ignorado (flag -b)."
161                             else
162                                 echo "cp -a -v \"\$file_path\" \"$bkp\""
163                                 var=1
164                             fi
165                         else
166                             echo "Arquivo \"\$file_name\" será ignorado (flag -r)."
167                         fi
168                     else
169                         # Se o ficheiro estiver no ficheiro da flag -b, então é ignorado (Flags: -c -b)
170                         checkFile "$blockedFiles" "$file_path" "$file_name"
171                         if [ ${?} -eq 1 ]; then
172                             echo "Arquivo \"\$file_name\" será ignorado (flag -b)."
173                         else
174                             echo "cp -a -v \"\$file_path\" \"$bkp\""
175                             var=1
176                         fi
177                     fi
178                 else
179                     if [ ${sr_flag} -eq 1 ]; then
180                         # Se o nome do ficheiro verificar a expressão regular, então copiamos (Flags: -c -r)
181                         if [ "$file_name" =~ $regexpr ]; then
182                             echo "cp -a -v \"\$file_path\" \"$bkp\""
183                             var=1
184                         else
185                             echo "Arquivo \"\$file_name\" será ignorado (flag -r)."
186                         fi
187                     else
188                         # (Flags: -c)
189                         echo "cp -a -v \"\$file_path\" \"$bkp\""
190                         var=1
191                     fi
192                 fi
193             fi

```

Figura 27 - Cópia de ficheiros com \${c_flag}=1

```

193         else
194             if [ ${sb_flag} -eq 1 ]; then
195                 if [ ${sr_flag} -eq 1 ]; then
196                     if [ "$file_name" =~ $regexpr ]; then
197                         # Mesmo que o ficheiro verifique a expressão regular, se estiver no ficheiro da flag -b, então é ignorado (Flags: -b -r)
198                         checkFile "$blockedFiles" "$file_path" "$file_name"
199                         if [ ${?} -eq 1 ]; then
200                             echo "Arquivo \"\$file_name\" será ignorado (flag -b)."
201                         else
202                             cp -a -v "$file_path" "$bkp"
203                             var=1
204                         fi
205                     else
206                         echo "Arquivo \"\$file_name\" será ignorado (flag -r)."
207                     fi
208                 else
209                     # Se o ficheiro estiver no ficheiro da flag -b, então é ignorado (Flags: -b)
210                     checkFile "$blockedFiles" "$file_path" "$file_name"
211                     if [ ${?} -eq 1 ]; then
212                         echo "Arquivo \"\$file_name\" será ignorado (flag -b)."
213                     else
214                         cp -a -v "$file_path" "$bkp"
215                         var=1
216                     fi
217                 fi
218             else
219                 if [ ${sr_flag} -eq 1 ]; then
220                     # Se o nome do ficheiro verificar a expressão regular, então copiamos (Flags: -r)
221                     if [ "$file_name" =~ $regexpr ]; then
222                         cp -a -v "$file_path" "$bkp"
223                         var=1
224                     else
225                         echo "Arquivo \"\$file_name\" será ignorado (flag -r)."
226                     fi
227                 else
228                     cp -a -v "$file_path" "$bkp"
229                     var=1
230                 fi
231             fi
232         fi

```

Figura 26 - Cópia de ficheiros com \${c_flag}=0

Em relação ao programa backup.sh, a cópia de ficheiros em backup_summary.sh é realizada de forma semelhante, com a implementação da função `checkExistence()` e das variáveis `$control` e `$var`, que nos vão permitir verificar se o ficheiro foi copiado ou apenas atualizado.

Verificações finais

Se a variável \$var se encontrar com o valor 0, tal como quando foi inicializada, significa que o ficheiro em

```
233 # Verificação para ver se o ficheiro não foi ignorado
234 if [[ $var -eq 1 ]]; then
235     # Verificação final para atribuir os valores certos a Updated e Copied
236     if [[ $control -eq 1 ]]; then
237         ((updated+=1))
238     else
239         ((copied+=1))
240         copied_size=$((copied_size + $(stat -c% "$file_path")))
241     fi
242 fi
243 # Verifica se o ficheiro em src é mais antigo do que em bkp
244 elif [[ "$file_path" -ot "$bkp/$file_name" ]]; then
245     echo "WARNING: backup entry $bkp/$file_name is newer than $file_path; Should not happen"
246     ((warnings+=1))
247 fi
248 fi
249 done
250 fi
251 shopt -u dotglob
252 shopt -u nullglob
```

Figura 28 - Verificações finais

questão foi ignorado. Caso contrário, de é verificado o valor da variável \$control que, caso o seu valor seja igual a 1, significa que o ficheiro em questão já se encontrava em \$bkp, logo, este apenas é atualizado, incrementando assim a variável \$updated. Se o ficheiro não se encontrasse em \$bkp, este era copiado e a variável \$copied era incrementada. A variável \$copied_size vai armazenar o tamanho total, em *bytes*, dos ficheiros copiados.

Caso o ficheiro em questão fosse mais recente em \$bkp do que em \$src iria ser impresso um *WARNING* no terminal e era também incrementado o valor da variável \$warnings.

Impressão do resultado

```
254 echo -e "While backing up $src: $errors Errors; $warnings Warnings; $updated Updated; $copied Copied (${copied_size}B); $deleted Deleted (${deleted_size}B)\n"
```

Figura 29 - Impressão do resultado

Antes do fim do programa, é impresso o sumário da execução do programa, apresentando o número de erros, o número de avisos, o número de ficheiros atualizados, o número de ficheiros copiados e o tamanho, em *bytes*, dos mesmos, e por fim o número de ficheiros apagados e também o tamanho, em *bytes*, dos mesmos.

Fim do programa

```
256 # Exibir "Fim do programa" apenas na primeira execução (não recursiva)
257 if [ $INITIAL_CALL -eq 1 ]; then
258     echo "Fim do programa."
259 fi
```

Figura 30 - Fim do programa

É impresso no terminal uma mensagem de fim de programa.

3.4 backup_check.sh

Inicialização e sources

```
1  #!/bin/bash
2
3  # Verifica se $INITIAL_CALL não está definida
4  # Ao usarmos export fazemos com que qualquer child process (chamada recursiva) consiga aceder ao valor definido na chamada inicial
5  if [ -z "$INITIAL_CALL" ]; then
6  |   export INITIAL_CALL=1 # Estamos na chamada inicial
7  else
8  |   export INITIAL_CALL=0 # Estamos na chamada recursiva
9  fi
10
11 source ./functions/checkPath.sh
12 source ./functions/isNewer.sh
13 source ./functions/checkFile.sh
```

Figura 31 - Inicialização e sources

A inicialização em *backup_check.sh* é muito semelhante à realizado em *backup.sh*. Verificamos se nos encontramos na chamada inicial, ou na recursiva e são também invocadas as funções que irão ser usadas no programa.

Asserções Iniciais

```
15 # $1 - src (Pasta a copiar)
16 # $2 - bkp (Pasta onde colar) [Criar Pasta caso não exista]
17
18 if [ $# -ne 2 ]; then
19 |   echo -e "Parâmetros incorretos!\nEsperado: /path/to/src /path/to/bkp"
20 |   exit 1
21 fi
22
23 # Atribuir os restantes argumentos a src e bkp
24 src="$1"
25 bkp="$2"
26
27 # Verificar se $src existe
28 checkPath "$src"
29 if [ $? -ne 1 ]; then
30 |   echo "O caminho \\"$src\\" não existe!"
31 |   exit 1;
32 fi
33
34 # Verificar se $bkp existe
35 checkPath "$bkp"
36 if [ $? -ne 1 ]; then
37 |   echo "O caminho \\"$bkp\\" não existe!"
38 |   exit 1;
39 fi
```

Figura 32 - Asserções Iniciais

O processo de início do programa é feito de forma semelhante ao que já foi efetuado em *backup.sh*. É-se verificado se o programa tem exatamente dois argumentos, são atribuídos às variáveis \$bkp e \$src os seus respetivos valores, e é-se também verificada a existência do caminho para a \$src e para \$bkp.

Entrada recursiva em sub-diretórios

```
42 shopt -s nullglob
43 shopt -s dotglob
44 for file_path in "$src"/*; do
45     file_name=${basename "$file_path"}      # Remove o prefixo do caminho e deixa apenas o nome do ficheiro/diretório
46
47     if [ -d "$file_path" ]; then
48         checkPath "$bkp/$file_name"
49         if [[ $? -eq 1 ]]; then
50             eval "bash \"\$0\" \"\$file_path\" \"\$bkp/\$file_name\""
51         fi
52     else
53         if [[ -e "$bkp/$file_name" ]]; then
54             # A função md5sum devolve um hash associado e o nome do ficheiro(se os hashes forem iguais então os ficheiros também são)
55             # awk '{print $1}' é usado para filtrar o nome do ficheiro e obter apenas o código hash
56
57             hash_src=$(md5sum "$file_path" | awk '{print $1}')
58             hash_bkp=$(md5sum "$bkp/$file_name" | awk '{print $1}')
59
60             if [ ! "$hash_src" == "$hash_bkp" ]; then
61                 echo "$file_path $bkp/$file_name differ."
62             fi
63         fi
64     fi
65 done
66 shopt -u dotglob
67 shopt -u nullglob
```

Figura 33 - Entrada recursiva em sub-diretórios

Para todos os diretórios encontrados no caminho fornecido é verificado se esse diretório existe em \$bkp e, caso isso se verifique, é chamado o *script* recursivamente no próprio diretório.

Comparação de ficheiros

```
52     else
53         if [[ -e "$bkp/$file_name" ]]; then
54             # A função md5sum devolve um hash associado e o nome do ficheiro(se os hashes forem iguais então os ficheiros também são)
55             # awk '{print $1}' é usado para filtrar o nome do ficheiro e obter apenas o código hash
56
57             hash_src=$(md5sum "$file_path" | awk '{print $1}')
58             hash_bkp=$(md5sum "$bkp/$file_name" | awk '{print $1}')
59
60             if [ ! "$hash_src" == "$hash_bkp" ]; then
61                 echo "$file_path $bkp/$file_name differ."
62             fi
63         fi
64     fi
65 done
66 shopt -u dotglob
67 shopt -u nullglob
```

Figura 34 - Comparação de ficheiros

Para os ficheiros encontrados no caminho fornecido verifica-se se o ficheiro também se encontra em \$bkp e, caso isso se verifique, o *script* verifica se o ficheiro no diretório original e o ficheiro em \$bkp são diferentes, comparando os seus valores de *hash*. Se isso se verificar, é impressa no terminal uma mensagem dizendo ao utilizador que os ficheiros diferem.

Fim do programa

```
69     # Exibir "Fim do programa" apenas na primeira execução (não recursiva)
70     if [ $INITIAL_CALL -eq 1 ]; then
71         echo "Fim do programa."
72     fi
```

Figura 35 - Fim do programa

No final, é impressa a mensagem de fim de programa.

6. Testes (*a priori*)

Estes testes foram efetuados antes de “BACKUP” ter sido criada, daí serem *a priori*.

6.1 backup_files.sh

SOURCE

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE$ ls -a
.  ..  file1.txt  file2.log  file3.c  file4.pdf  .hidden1  .hidden2  'just      another space.html'  'literally some      space'
```

Figura 36 - Organização interna de SOURCE

Testes Válidos

Teste 1 – Sem flags

```
./backup_files.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_files.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
'/home/tiago/Desktop/SOURCE/file1.txt' -> '/home/tiago/Desktop/BACKUP/file1.txt'
'/home/tiago/Desktop/SOURCE/file2.log' -> '/home/tiago/Desktop/BACKUP/file2.log'
'/home/tiago/Desktop/SOURCE/file3.c' -> '/home/tiago/Desktop/BACKUP/file3.c'
'/home/tiago/Desktop/SOURCE/file4.pdf' -> '/home/tiago/Desktop/BACKUP/file4.pdf'
'/home/tiago/Desktop/SOURCE/.hidden1' -> '/home/tiago/Desktop/BACKUP/.hidden1'
'/home/tiago/Desktop/SOURCE/.hidden2' -> '/home/tiago/Desktop/BACKUP/.hidden2'
'/home/tiago/Desktop/SOURCE/just      another space.html' -> '/home/tiago/Desktop/BACKUP/just      another space.html'
'/home/tiago/Desktop/SOURCE/literally some      space' -> '/home/tiago/Desktop/BACKUP/literally some      space'
Fim do programa.
```

Figura 37 - Teste 1



Como esperado, ao não utilizarmos nenhuma *flag*, o programa copia corretamente todos os ficheiros de SOURCE para BACKUP, incluindo ficheiros escondidos (“.hidden1” e “.hidden2”) e ficheiros cujo nome incluem espaços (“literally some space” e “just another space.html”).

Teste 2 – Com flag -c

```
./backup_files.sh -c /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_files.sh -c /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir -p -v "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file1.txt" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file2.log" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file3.c" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file4.pdf" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/.hidden1" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/.hidden2" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/just      another space.html" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/literally some      space" "/home/tiago/Desktop/BACKUP"
Fim do programa.
```

Figura 38 - Teste 2



Como esperado, ao utilizarmos a *flag* -c, o programa simula corretamente a cópia de todos os ficheiros de SOURCE para BACKUP, incluindo ficheiros escondidos (“.hidden1” e “.hidden2”) e ficheiros cujo nome incluem espaços (“literally some space” e “just another space.html”).

Testes Inválidos

Teste 3 - Path de Source Inexistente

```
./backup_files.sh -c /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup_files.sh /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
0 caminho "/home/tiago/Desktop/NOTHING" não existe!
```

Figura 39 - Teste 3



Como esperado, ao utilizarmos uma SOURCE inválida (“/home/tiago/Desktop/NOTHING”), o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Teste 4 – Path de Source e Backup iguais

```
./backup_files.sh -c /home/tiago/Desktop/NOTHING /home/tiago/Desktop/SOURCE
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup_files.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/SOURCE
Os caminhos "/path/to/src" e "/path/to/bkp" não podem ser iguais!
```

Figura 40 - Teste 4



Como esperado, ao utilizarmos um BACKUP igual a SOURCE, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Teste 5 – Com flag -c e argumentos para -c

```
./backup_files.sh -c some_argument /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup_files.sh -c some_argument /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
Parâmetros incorretos!
Esperado: -c /path/to/src /path/to/bkp
```

Figura 41 - Teste 5



Como esperado, ao utilizarmos um argumento (“some_argument”) em conjunto com a flag -c, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro e ainda explica como deve ser feita a introdução de argumentos.

6.2 backup.sh

SOURCE

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE$ ls -a
. .. DirA DirB file1.txt file2.log file3.c .hidden1 'literally some      space'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE$ cd DirA
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA$ ls -a
. .. DirC file4.exe file5.html file6.exe .hidden2 'just      another space.csv'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA$ cd DirC
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA/DirC$ ls -a
. .. '.single file'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA/DirC$ cd ../../DirB
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirB$ ls -a
. .. file7.md file8.ps file9.pdf 'final      space' .hidden3
```

Figura 42 - Organização interna de SOURCE

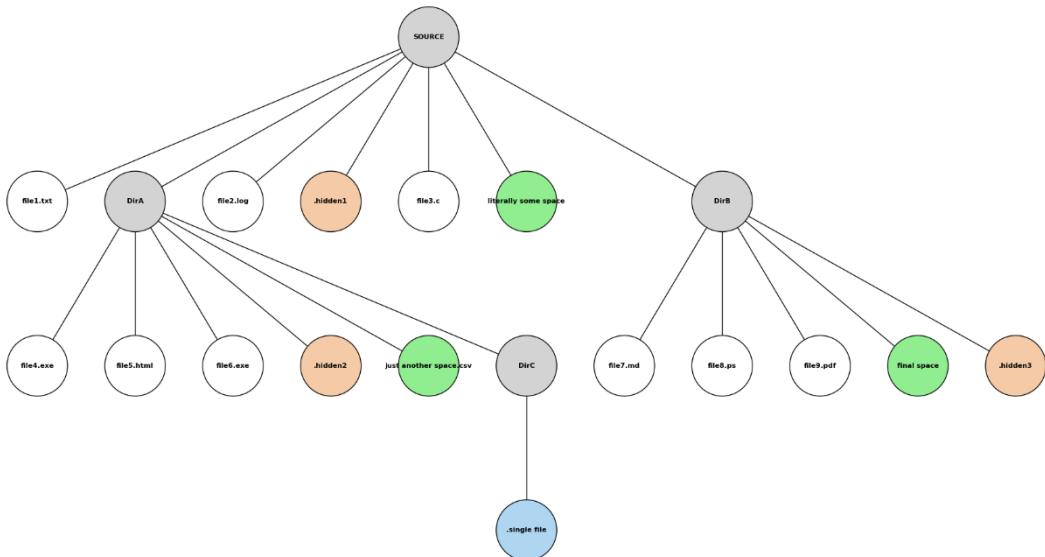


Figura 43 – Grafo de SOURCE

exclude_list.txt

O conteúdo do arquivo 'exclude_list.txt' é:

```
some_random_file1.txt
/home/tiago/Desktop/SOURCE/DirA/.hidden2
some_random_file2.txt

some_random_file3.txt
just      another space.csv
file7.md

file9.pdf
```

Figura 43 - Conteúdo de "exclude_list.txt"

Testes Válidos

Teste 1 – Sem flags

```
./backup.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA/DirC'
'/home/tiago/Desktop/SOURCE/DirA/DirC/.single_file' -> '/home/tiago/Desktop/BACKUP/DirA/DirC/.single_file'
'/home/tiago/Desktop/SOURCE/DirA/file4.exe' -> '/home/tiago/Desktop/BACKUP/DirA/file4.exe'
'/home/tiago/Desktop/SOURCE/DirA/file5.html' -> '/home/tiago/Desktop/BACKUP/DirA/file5.html'
'/home/tiago/Desktop/SOURCE/DirA/file6.exe' -> '/home/tiago/Desktop/BACKUP/DirA/file6.exe'
'/home/tiago/Desktop/SOURCE/DirA/.hidden2' -> '/home/tiago/Desktop/BACKUP/DirA/.hidden2'
'/home/tiago/Desktop/SOURCE/DirA/just another space.csv' -> '/home/tiago/Desktop/BACKUP/DirA/just another space.csv'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirB'
'/home/tiago/Desktop/SOURCE/DirB/file7.md' -> '/home/tiago/Desktop/BACKUP/DirB/file7.md'
'/home/tiago/Desktop/SOURCE/DirB/file8.ps' -> '/home/tiago/Desktop/BACKUP/DirB/file8.ps'
'/home/tiago/Desktop/SOURCE/DirB/file9.pdf' -> '/home/tiago/Desktop/BACKUP/DirB/file9.pdf'
'/home/tiago/Desktop/SOURCE/DirB/final space' -> '/home/tiago/Desktop/BACKUP/DirB/final space'
'/home/tiago/Desktop/SOURCE/DirB/.hidden3' -> '/home/tiago/Desktop/BACKUP/DirB/.hidden3'
'/home/tiago/Desktop/SOURCE/file1.txt' -> '/home/tiago/Desktop/BACKUP/file1.txt'
'/home/tiago/Desktop/SOURCE/file2.log' -> '/home/tiago/Desktop/BACKUP/file2.log'
'/home/tiago/Desktop/SOURCE/file3.c' -> '/home/tiago/Desktop/BACKUP/file3.c'
'/home/tiago/Desktop/SOURCE/.hidden1' -> '/home/tiago/Desktop/BACKUP/.hidden1'
'/home/tiago/Desktop/SOURCE/literally some space' -> '/home/tiago/Desktop/BACKUP/literally some space'
Fim do programa.
```

Figura 45 - Teste 1

Como esperado, ao não utilizarmos nenhuma *flag*, o programa copia corretamente todos os ficheiros e sub-diretorias de SOURCE para BACKUP, incluindo ficheiros escondidos (“.hidden1”, “.hidden2”, “.hidden3” e “.single_file”) e ficheiros cujo nome incluem espaços (“literally some space”, “just another space.html” e “final space”).



Teste 2 – Com flag -c

```
./backup.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup.sh -c /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir -p -v "/home/tiago/Desktop/BACKUP"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
eval "bash ./backup.sh" -c "/home/tiago/Desktop/SOURCE/DirA" "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
eval "bash ./backup.sh" -c "/home/tiago/Desktop/SOURCE/DirA/DirC" "/home/tiago/Desktop/BACKUP/DirA/DirC"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/DirC/.single_file" "/home/tiago/Desktop/BACKUP/DirA/DirC"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file4.exe" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file5.html" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file6.exe" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/.hidden2" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/just another space.csv" "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
eval "bash ./backup.sh" -c "/home/tiago/Desktop/SOURCE/DirB" "/home/tiago/Desktop/BACKUP/DirB"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/file7.md" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/file8.ps" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/file9.pdf" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/final space" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/.hidden3" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/file1.txt" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file2.log" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file3.c" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/.hidden1" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/literally some space" "/home/tiago/Desktop/BACKUP"
Fim do programa.
```

Figura 46 - Teste 2



Como esperado, ao utilizarmos a flag `-c`, o programa simula corretamente a cópia de todos os ficheiros e sub-diretorias de SOURCE para BACKUP, incluindo ficheiros escondidos (`.hidden1`, `.hidden2`, `.hidden3` e `.single_file`) e ficheiros cujo nome incluem espaços (`literally some space`, `just another space.html` e `final space`).

Teste 3 – Com flag -b

```
./backup.sh -b /home/tiago/Desktop/Teste/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup.sh -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA/DirC'
'/home/tiago/Desktop/SOURCE/DirA/DirC/.single_file' -> '/home/tiago/Desktop/BACKUP/DirA/DirC/.single_file'
'/home/tiago/Desktop/SOURCE/DirA/File4.exe' -> '/home/tiago/Desktop/BACKUP/DirA/File4.exe'
'/home/tiago/Desktop/SOURCE/DirA/File5.html' -> '/home/tiago/Desktop/BACKUP/DirA/File5.html'
'/home/tiago/Desktop/SOURCE/DirA/File6.exe' -> '/home/tiago/Desktop/BACKUP/DirA/File6.exe'
Arquivo ".hidden2" será ignorado (flag -b).
Arquivo "just another space.csv" será ignorado (flag -b).
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirB'
Arquivo "file7.md" será ignorado (flag -b).
'/home/tiago/Desktop/SOURCE/DirB/File8.ps' -> '/home/tiago/Desktop/BACKUP/DirB/file8.ps'
Arquivo "file9.pdf" será ignorado (flag -b).
'/home/tiago/Desktop/SOURCE/DirB/Final      space' -> '/home/tiago/Desktop/BACKUP/DirB/final      space'
'/home/tiago/Desktop/SOURCE/DirB/hidden3' -> '/home/tiago/Desktop/BACKUP/DirB/hidden3'
'/home/tiago/Desktop/SOURCE/file1.txt' -> '/home/tiago/Desktop/BACKUP/file1.txt'
'/home/tiago/Desktop/SOURCE/file2.log' -> '/home/tiago/Desktop/BACKUP/file2.log'
'/home/tiago/Desktop/SOURCE/file3.c' -> '/home/tiago/Desktop/BACKUP/file3.c'
'/home/tiago/Desktop/SOURCE/.hidden1' -> '/home/tiago/Desktop/BACKUP/.hidden1'
'/home/tiago/Desktop/SOURCE/literally some   space' -> '/home/tiago/Desktop/BACKUP/literally some   space'
Fim do programa.
```

Figura 47 - Teste 3



Como esperado, ao utilizarmos a flag `-b` e, como seu argumento “`exclude_list.txt`”, o programa copia corretamente todos os ficheiros e sub-diretorias de SOURCE para BACKUP, incluindo ficheiros escondidos (“`.hidden1`”, “`.hidden3`” e “`.single_file`”) e ficheiros cujo nome incluem espaços (“literally some space” e “final space”).

Os únicos ficheiros que não são copiados são aqueles cujos caminhos/nomes foram encontrados em “`exclude_list.txt`”, esses sendo (“`.hidden2`”, “`just another space.csv`”, “`file7.md`” e “`file9.pdf`”).

Teste 4 – Com flag -r

```
./backup.sh -r '.*\.txt$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup.sh -r '.*\.txt$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA/DirC'
Arquivo ".single_file" será ignorado (flag -r).
Arquivo "file4.exe" será ignorado (flag -r).
Arquivo "file5.html" será ignorado (flag -r).
Arquivo "file6.exe" será ignorado (flag -r).
Arquivo ".hidden2" será ignorado (flag -r).
Arquivo "just another space.csv" será ignorado (flag -r).
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirB'
Arquivo "file7.md" será ignorado (flag -r).
Arquivo "file8.ps" será ignorado (flag -r).
Arquivo "file9.pdf" será ignorado (flag -r).
Arquivo "final      space" será ignorado (flag -r).
Arquivo ".hidden3" será ignorado (flag -r).
'/home/tiago/Desktop/SOURCE/file1.txt' -> '/home/tiago/Desktop/BACKUP/file1.txt'
Arquivo "file2.log" será ignorado (flag -r).
Arquivo "file3.c" será ignorado (flag -r).
Arquivo ".hidden1" será ignorado (flag -r).
Arquivo "literally some   space" será ignorado (flag -r).
Fim do programa.
```

Figura 448 - Teste 4



Como esperado, ao utilizarmos a flag `-r` e, como seu argumento “`.*\.txt$`”, o programa copia corretamente apenas um ficheiro, sendo esse “`file1.txt`”, uma vez que termina em ‘.txt’, verificando assim a expressão regular pretendida.

Teste 5 – Com flags -c e -b

```
./backup.sh -c -b /home/tiago/Desktop/Teste/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:/Desktop/Trabalho-1---50---2024-23$ ./backup.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir -p -v "/home/tiago/Desktop/BACKUP"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
eval "bash ./backup.sh" -c -b /home/tiago/Desktop/Testes/exclude_list.txt "/home/tiago/Desktop/SOURCE/DirA" "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
eval "bash ./backup.sh" -c -b /home/tiago/Desktop/Testes/exclude_list.txt "/home/tiago/Desktop/SOURCE/DirA/DirC" "/home/tiago/Desktop/BACKUP/DirA/DirC"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/.single_file" "/home/tiago/Desktop/BACKUP/DirA/DirC"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file4.exe" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/files5.html" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file6.exe" "/home/tiago/Desktop/BACKUP/DirA"
Arquivo ".hidden2" será ignorado (flag -b).
Arquivo "just another space.csv" será ignorado (flag -b).
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
eval "bash ./backup.sh" -c -b /home/tiago/Desktop/Testes/exclude_list.txt "/home/tiago/Desktop/SOURCE/DirB" "/home/tiago/Desktop/BACKUP/DirB"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
Arquivo "file7.md" será ignorado (flag -b).
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/file8.ps" "/home/tiago/Desktop/BACKUP/DirB"
Arquivo "file9.pdf" será ignorado (flag -b).
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/final      space" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/.hidden3" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/file1.txt" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file2.log" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file3.c" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/.hidden1" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/literally some   space" "/home/tiago/Desktop/BACKUP"
Fim do programa.
```

Figura 459 - Teste 5



Como esperado, ao utilizarmos a *flag* -c e a *flag* -b e, como seu argumento “*exclude_list.txt*”, o programa simula corretamente a cópia de todos os ficheiros e sub-diretorias de SOURCE para BACKUP, incluindo ficheiros escondidos (“*.hidden1*”, “*.hidden3*” e “*.single_file*”) e ficheiros cujo nome incluem espaços (“*literally some space*” e “*final space*”).

Os únicos ficheiros que não são copiados são aqueles cujos caminhos/nomes foram encontrados em “*exclude_list.txt*”, esses sendo (“*.hidden2*”, “*just another space.csv*”, “*file7.md*” e “*file9.pdf*”).

Teste 6 – Com flags -b e -r

```
./backup.sh -r '.*\.log$' -b /home/tiago/Desktop/Teste/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:/Desktop/Trabalho-1---50---2024-23$ ./backup.sh -r '.*\.log$' -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA/DirC'
Arquivo ".single_file" será ignorado (flag -r).
Arquivo "file4.exe" será ignorado (flag -r).
Arquivo "file5.html" será ignorado (flag -r).
Arquivo "file6.exe" será ignorado (flag -r).
Arquivo ".hidden2" será ignorado (flag -r).
Arquivo "just another space.csv" será ignorado (flag -r).
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirB'
Arquivo "file7.md" será ignorado (flag -r).
Arquivo "file8.ps" será ignorado (flag -r).
Arquivo "file9.pdf" será ignorado (flag -r).
Arquivo "final      space" será ignorado (flag -r).
Arquivo ".hidden3" será ignorado (flag -r).
Arquivo "file1.txt" será ignorado (flag -r).
'./home/tiago/Desktop/SOURCE/file2.log' -> '/home/tiago/Desktop/BACKUP/file2.log'
Arquivo "file3.c" será ignorado (flag -r).
Arquivo ".hidden1" será ignorado (flag -r).
Arquivo "literally some   space" será ignorado (flag -r).
Fim do programa.
```

Figura 50 - Teste 6



Como esperado, ao utilizarmos as *flags* -b e -r e, como seus argumentos “*exclude_list.txt*” e “*.*\.log\$*”, respetivamente, o programa copia corretamente apenas um ficheiro, sendo esse “*file2.log*”, uma vez que termina em ‘.log’, verificando assim a expressão regular pretendida e o seu caminho/nome não foi encontrado em “*exclude_list.txt*”.

Teste 7 – Com flags -c, -b e -r

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt -r '.*\.txt$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir -p -v "/home/tiago/Desktop/BACKUP"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
eval "bash ./backup.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt -r '.*\.txt$' "/home/tiago/Desktop/SOURCE/DirA" "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
eval "bash ./backup.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt -r '.*\.txt$' "/home/tiago/Desktop/SOURCE/DirA/DirC" "/home/tiago/Desktop/BACKUP/DirA/DirC"
mkdir -p -v "/home/tiago/Desktop/DirA/DirC"
Arquivo "single file" será ignorado (flag -r).
Arquivo "file4.exe" será ignorado (flag -r).
Arquivo "file5.html" será ignorado (flag -r).
Arquivo "file6.exe" será ignorado (flag -r).
Arquivo ".hidden2" será ignorado (flag -r).
Arquivo "just another space.csv" será ignorado (flag -r).
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
eval "bash ./backup.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt -r '.*\.txt$' "/home/tiago/Desktop/SOURCE/DirB" "/home/tiago/Desktop/BACKUP/DirB"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
Arquivo "file7.md" será ignorado (flag -r).
Arquivo "file8.ps" será ignorado (flag -r).
Arquivo "file9.pdf" será ignorado (flag -r).
Arquivo "final space" será ignorado (flag -r).
Arquivo ".hidden3" será ignorado (flag -r).
cp -a -v "/home/tiago/Desktop/SOURCE/file1.txt" /home/tiago/Desktop/BACKUP
Arquivo "file2.log" será ignorado (flag -r).
Arquivo "file3.c" será ignorado (flag -r).
Arquivo ".hidden1" será ignorado (flag -r).
Arquivo "literally some space" será ignorado (flag -r).
Fim do programa.
```

Figura 51 - Teste 7



Como esperado, ao utilizarmos a flag -c e as flags -b e -r e, como seus argumentos “exclude_list.txt” e “.*\.txt\$”, respectivamente, o programa simula corretamente a cópia de apenas um ficheiro, sendo esse “file1.txt”, uma vez que termina em ‘.txt’, verificando assim a expressão regular pretendida e o seu caminho/nome não foi encontrado em “exclude_list.txt”.

Testes Inválidos

Teste 8 – Path de Source Inexistente

```
./backup.sh /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup.sh /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
0 caminho "/home/tiago/Desktop/NOTHING" não existe!
```

Figura 52 - Teste 8



Como esperado, ao utilizarmos uma SOURCE inválida (“/home/tiago/Desktop/NOTHING”), o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Teste 9 – Path de Source e Backup iguais

```
./backup.sh -c /home/tiago/Desktop/SOURCE /home/tiago/Desktop/SOURCE
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/SOURCE
Os caminhos "/path/to/src" e "/path/to/bkp" não podem ser iguais!
```

Figura 53 - Teste 9



Como esperado, ao utilizarmos um BACKUP igual a SOURCE, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Teste 10 – Com flag -c e argumentos para -c

```
./backup.sh -c some_argument /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup.sh -c some_argument /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
Parâmetros incorretos!
Esperado: -c -b [tfile] -r [regexp] /path/to/src /path/to/bkp
```

Figura 46 - Teste 10



Como esperado, ao utilizarmos um argumento (“some_argument”) em conjunto com a *flag* -c, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro e ainda explica como deve ser feita a introdução de argumentos.

Teste 11 – Com flag -b e "exclude_list.txt" inválido/não existente

```
./backup.sh -b /home/tiago/Desktop/Testes/non_existent.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup.sh -b /home/tiago/Desktop/Testes/non_existent.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
[tfile] tem de ser um ficheiro válido!
```

Figura 55 - Teste 11



Como esperado, ao utilizarmos um argumento inexistente (“/home/tiago/Desktop/Testes/non_existent.txt”) em conjunto com a *flag* -b, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

6.3 backup_summary.sh

SOURCE

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE$ ls -a
. .. DirA DirB file1.txt file2.log file3.c .hidden1 'literally some      space'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE$ cd DirA
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA$ ls -a
. .. DirC file4.exe file5.html file6.exe .hidden2 'just      another space.csv'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA$ cd DirC
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA/DirC$ ls -a
. .. '.single file'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA/DirC$ cd ../../DirB
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirB$ ls -a
. .. file7.md file8.ps file9.pdf 'final      space' .hidden3
```

Figura 56 – Organização interna de SOURCE

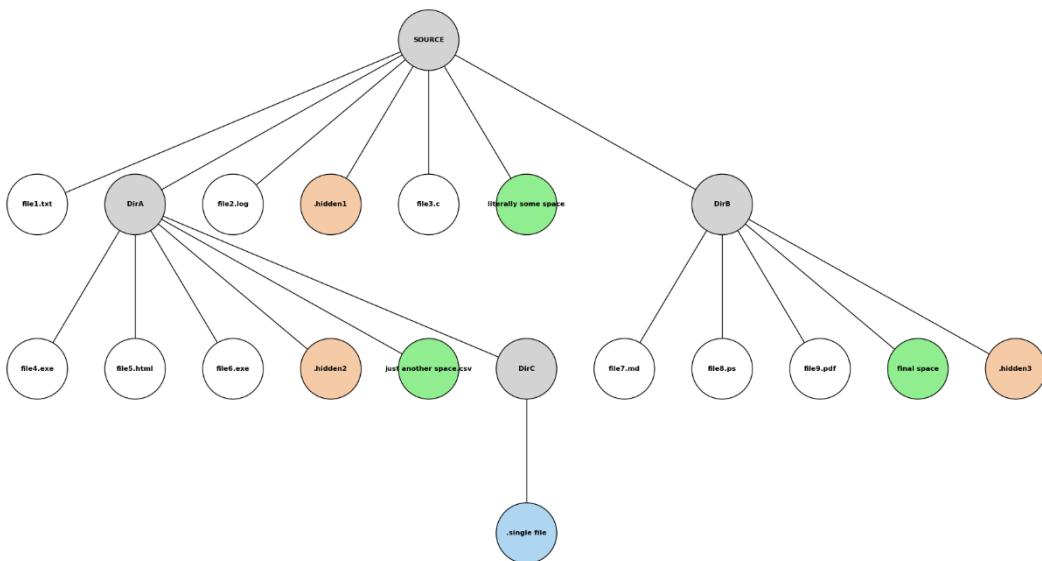


Figura 57 – Grafo de SOURCE

exclude_list.txt

```
some_random_file1.txt
/home/tiago/Desktop/SOURCE/DirA/.hidden2
some_random_file2.txt

some_random_file3.txt
just      another space.csv
file7.md

file9.pdf
```

Figura 58 - Conteúdo de "exclude_list.txt"

Testes Válidos

Teste 1 – Sem flags

```
./backup_summary.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA/DirC'
'./home/tiago/Desktop/SOURCE/DirA/DirC/.single_file' -> './home/tiago/Desktop/BACKUP/DirA/DirC/.single_file'
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (0B); 0 Deleted (0B)

'./home/tiago/Desktop/SOURCE/DirA/file4.exe' -> './home/tiago/Desktop/BACKUP/DirA/file4.exe'
'./home/tiago/Desktop/SOURCE/DirA/file5.html' -> './home/tiago/Desktop/BACKUP/DirA/file5.html'
'./home/tiago/Desktop/SOURCE/DirA/file6.exe' -> './home/tiago/Desktop/BACKUP/DirA/file6.exe'
'./home/tiago/Desktop/SOURCE/DirA/.hidden2' -> './home/tiago/Desktop/BACKUP/DirA/.hidden2'
'./home/tiago/Desktop/SOURCE/DirA/just another space.csv' -> './home/tiago/Desktop/BACKUP/DirA/just another space.csv'
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 0 Updated; 5 Copied (0B); 0 Deleted (0B)

mkdir: created directory '/home/tiago/Desktop/BACKUP/DirB'
'./home/tiago/Desktop/SOURCE/DirB/file7.md' -> './home/tiago/Desktop/BACKUP/DirB/file7.md'
'./home/tiago/Desktop/SOURCE/DirB/file8.ps' -> './home/tiago/Desktop/BACKUP/DirB/file8.ps'
'./home/tiago/Desktop/SOURCE/DirB/file9.pdf' -> './home/tiago/Desktop/BACKUP/DirB/file9.pdf'
'./home/tiago/Desktop/SOURCE/DirB/final space' -> './home/tiago/Desktop/BACKUP/DirB/final space'
'./home/tiago/Desktop/SOURCE/DirB/.hidden3' -> './home/tiago/Desktop/BACKUP/DirB/.hidden3'
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 5 Copied (0B); 0 Deleted (0B)

'./home/tiago/Desktop/SOURCE/file1.txt' -> './home/tiago/Desktop/BACKUP/file1.txt'
'./home/tiago/Desktop/SOURCE/file2.log' -> './home/tiago/Desktop/BACKUP/file2.log'
'./home/tiago/Desktop/SOURCE/file3.c' -> './home/tiago/Desktop/BACKUP/file3.c'
'./home/tiago/Desktop/SOURCE/.hidden1' -> './home/tiago/Desktop/BACKUP/.hidden1'
'./home/tiago/Desktop/SOURCE/literally some space' -> './home/tiago/Desktop/BACKUP/literally some space'
'./home/tiago/Desktop/SOURCE/just another space.html' -> './home/tiago/Desktop/BACKUP/just another space.html'
'./home/tiago/Desktop/SOURCE/final space' -> './home/tiago/Desktop/BACKUP/final space'
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 5 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Figura 59 – Teste1



Como esperado, ao não utilizarmos nenhuma *flag*, o programa copia corretamente todos os ficheiros e sub-diretorias de SOURCE para BACKUP, incluindo ficheiros escondidos (“.hidden1”, “.hidden2”, “.hidden3” e “.single_file”) e ficheiros cujo nome incluem espaços (“literally some space”, “just another space.html” e “final space”).

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Teste 2 – Com flag -c

```
./backup_summary.sh -c /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh -c /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir -p -v "/home/tiago/Desktop/BACKUP"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
eval "bash ./backup_summary.sh" -c "/home/tiago/Desktop/SOURCE/DirA" "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
eval "bash ./backup_summary.sh" -c "/home/tiago/Desktop/SOURCE/DirA/DirC" "/home/tiago/Desktop/BACKUP/DirA/DirC"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/DirC/.single_file" "/home/tiago/Desktop/BACKUP/DirA/DirC"
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (0B); 0 Deleted (0B)

cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file4.exe" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file5.html" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file6.exe" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/.hidden2" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/just another space.csv" "/home/tiago/Desktop/BACKUP/DirA"
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 0 Updated; 5 Copied (0B); 0 Deleted (0B)

mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
eval "bash ./backup_summary.sh" -c "/home/tiago/Desktop/SOURCE/DirB" "/home/tiago/Desktop/BACKUP/DirB"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/file7.md" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/file8.ps" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/file9.pdf" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/final space" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/.hidden3" "/home/tiago/Desktop/BACKUP/DirB"
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 5 Copied (0B); 0 Deleted (0B)

cp -a -v "/home/tiago/Desktop/SOURCE/file1.txt" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file2.log" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file3.c" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/.hidden1" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/literally some space" "/home/tiago/Desktop/BACKUP"
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 5 Copied (0B); 0 Deleted (0B)
```

Fim do programa.

Figura 60 – Teste 2



Como esperado, ao utilizarmos a *flag* -c, o programa simula corretamente a cópia de todos os ficheiros e sub-diretorias de SOURCE para BACKUP, incluindo ficheiros escondidos (“.hidden1”, “.hidden2”, “.hidden3” e “.single_file”) e ficheiros cujo nome incluem espaços (“literally some space”, “just another space.html” e “final space”).

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Teste 3 – Com flag -b

```
./backup_summary.sh -b /home/tiago/Desktop/Testes/exclude_file.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP

tiago@tiago-IdeaPad-3-15ALC6:/Desktop/Trabalho-1---50---2024-25$ ./backup_summary.sh -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA/DirC'
'/home/tiago/Desktop/SOURCE/DirA/DirC/.single_file' -> '/home/tiago/Desktop/BACKUP/DirA/DirC/.single_file'
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/DirA/file4.exe' -> '/home/tiago/Desktop/BACKUP/DirA/file4.exe'
'/home/tiago/Desktop/SOURCE/DirA/file5.html' -> '/home/tiago/Desktop/BACKUP/DirA/file5.html'
'/home/tiago/Desktop/SOURCE/DirA/file6.exe' -> '/home/tiago/Desktop/BACKUP/DirA/file6.exe'
Arquivo ".hidden2" será ignorado (flag -b).
Arquivo "just another space.csv" será ignorado (flag -b).
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 0 Updated; 3 Copied (0B); 0 Deleted (0B)

mkdir: created directory '/home/tiago/Desktop/BACKUP/DirB'
Arquivo "file7.md" será ignorado (flag -b).
'/home/tiago/Desktop/SOURCE/DirB/file8.ps' -> '/home/tiago/Desktop/BACKUP/DirB/file8.ps'
Arquivo "file9.pdf" será ignorado (flag -b).
'/home/tiago/Desktop/SOURCE/DirB/final      space' -> '/home/tiago/Desktop/BACKUP/DirB/final      space'
'/home/tiago/Desktop/SOURCE/DirB/.hidden3' -> '/home/tiago/Desktop/BACKUP/DirB/.hidden3'
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 3 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/file1.txt' -> '/home/tiago/Desktop/BACKUP/file1.txt'
'/home/tiago/Desktop/SOURCE/file2.log' -> '/home/tiago/Desktop/BACKUP/file2.log'
'/home/tiago/Desktop/SOURCE/file3.c' -> '/home/tiago/Desktop/BACKUP/file3.c'
'/home/tiago/Desktop/SOURCE/.hidden1' -> '/home/tiago/Desktop/BACKUP/.hidden1'
'/home/tiago/Desktop/SOURCE/literally some      space' -> '/home/tiago/Desktop/BACKUP/literally some      space'
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 5 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Figura 61 - Teste 3

Como esperado, ao utilizarmos a flag `-b` e, como seu argumento “`exclude_list.txt`”, o programa copia corretamente todos os ficheiros e sub-diretorias de `SOURCE` para `BACKUP`, incluindo ficheiros escondidos (“`.hidden1`”, “`.hidden3`” e “`.single_file`”) e ficheiros cujo nome incluem espaços (“`literally some space`” e “`final space`”).

Os únicos ficheiros que não são copiados são aqueles cujos caminhos/nomes foram encontrados em “`exclude_list.txt`”, esses sendo (“`.hidden2`”, “`just another space.csv`”, “`file7.md`” e “`file9.pdf`”).

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.



Teste 4 – Com flag -r

```
./backup_summary.sh -r '.*\.txt$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup_summary.sh -r '.*\.txt$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA/DirC'
Arquivo ".single file" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Arquivo "file4.exe" será ignorado (flag -r).
Arquivo "file5.html" será ignorado (flag -r).
Arquivo "file6.exe" será ignorado (flag -r).
Arquivo ".hidden2" será ignorado (flag -r).
Arquivo "just another space.csv" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

mkdir: created directory '/home/tiago/Desktop/BACKUP/DirB'
Arquivo "file7.md" será ignorado (flag -r).
Arquivo "file8.ps" será ignorado (flag -r).
Arquivo "file9.pdf" será ignorado (flag -r).
Arquivo "final space" será ignorado (flag -r).
Arquivo ".hidden3" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/file1.txt' -> '/home/tiago/Desktop/BACKUP/file1.txt'
Arquivo "file2.log" será ignorado (flag -r).
Arquivo "file3.c" será ignorado (flag -r).
Arquivo ".hidden1" será ignorado (flag -r).
Arquivo "literally some space" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Figura 62 – Teste 4

Como esperado, ao utilizarmos a flag `-r` e, como seu argumento `'.*\.txt$'`, o programa copia corretamente apenas um ficheiro, sendo esse “`file1.txt`”, uma vez que termina em `.txt`, verificando assim a expressão regular pretendida.

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.



Teste 5 – Com flags -c e -b

```
./backup_summary.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP

tiago@tiago-IdeaPad-3-15ALC6:/Desktop/Trabalho-1---50---2024-25$ ./backup_summary.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir -p -v "/home/tiago/Desktop/BACKUP"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
eval "bash "../backup_summary.sh" -c -b /home/tiago/Desktop/Testes/exclude_list.txt "/home/tiago/Desktop/SOURCE/DirA" "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
eval "bash "../backup_summary.sh" -c -b /home/tiago/Desktop/Testes/exclude_list.txt "/home/tiago/Desktop/SOURCE/DirA/DirC" "/home/tiago/Desktop/BACKUP/DirA/DirC"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/DirC/.single_file" "/home/tiago/Desktop/BACKUP/DirA/DirC"
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (0B); 0 Deleted (0B)

cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file4.exe" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/files.html" "/home/tiago/Desktop/BACKUP/DirA"
cp -a -v "/home/tiago/Desktop/SOURCE/DirA/file6.exe" "/home/tiago/Desktop/BACKUP/DirA"
Arquivo ".hidden2" será ignorado (flag -b).
Arquivo "just another space.csv" será ignorado (flag -b).
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 0 Updated; 3 Copied (0B); 0 Deleted (0B)

mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
eval "bash "../backup_summary.sh" -c -b /home/tiago/Desktop/Testes/exclude_list.txt "/home/tiago/Desktop/SOURCE/DirB" "/home/tiago/Desktop/BACKUP/DirB"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
Arquivo "file7.md" será ignorado (flag -b).
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/file8.ps" "/home/tiago/Desktop/BACKUP/DirB"
Arquivo "file9.pdf" será ignorado (flag -b).
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/final      space" "/home/tiago/Desktop/BACKUP/DirB"
cp -a -v "/home/tiago/Desktop/SOURCE/DirB/.hidden3" "/home/tiago/Desktop/BACKUP/DirB"
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 3 Copied (0B); 0 Deleted (0B)

cp -a -v "/home/tiago/Desktop/SOURCE/file1.txt" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file2.log" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/file3.c" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/.hidden1" "/home/tiago/Desktop/BACKUP"
cp -a -v "/home/tiago/Desktop/SOURCE/literally some   space" "/home/tiago/Desktop/BACKUP"
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 5 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Figura 63 – Teste 5



Como esperado, ao utilizarmos a flag `-c` e a flag `-b` e, como seu argumento “`exclude_list.txt`”, o programa simula corretamente a cópia de todos os ficheiros e sub-diretorias de SOURCE para BACKUP, incluindo ficheiros escondidos (“`.hidden1`”, “`.hidden3`” e “`.single_file`”) e ficheiros cujo nome incluem espaços (“`literally some space`” e “`final space`”).

Os únicos ficheiros que não são copiados são aqueles cujos caminhos/nomes foram encontrados em “`exclude_list.txt`”, esses sendo (“`.hidden2`”, “`just another space.csv`”, “`file7.md`” e “`file9.pdf`”).

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Teste 6 – Com flags -b e -r

```
./backup_summary.sh -r '.*\.log$' -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup_summary.sh -r '.*\.log$' -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir: created directory '/home/tiago/Desktop/BACKUP'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA'
mkdir: created directory '/home/tiago/Desktop/BACKUP/DirA/DirC'
Arquivo ".single file" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Arquivo "file4.exe" será ignorado (flag -r).
Arquivo "file5.html" será ignorado (flag -r).
Arquivo "file6.exe" será ignorado (flag -r).
Arquivo ".hidden" será ignorado (flag -r).
Arquivo "just another space.csv" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

mkdir: created directory '/home/tiago/Desktop/BACKUP/DirB'
Arquivo "file7.md" será ignorado (flag -r).
Arquivo "file8.ps" será ignorado (flag -r).
Arquivo "file9.pdf" será ignorado (flag -r).
Arquivo "final space" será ignorado (flag -r).
Arquivo ".hidden" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Arquivo "file1.txt" será ignorado (flag -r).
'/home/tiago/Desktop/SOURCE/file2.log' -> '/home/tiago/Desktop/BACKUP/file2.log'
Arquivo "file3.c" será ignorado (flag -r).
Arquivo ".hidden" será ignorado (flag -r).
Arquivo 'literally some space' será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Figura 64 – Teste 6



Como esperado, ao utilizarmos as *flags* -b e -r e, como seus argumentos “*exclude_list.txt*” e “*.*\.log\$*”, respetivamente, o programa copia corretamente apenas um ficheiro, sendo esse “*file2.log*”, uma vez que termina em ‘.log’, verificando assim a expressão regular pretendida e o seu caminho/nome não foi encontrado em “*exclude_list.txt*”.

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Teste 7 – Com flags -c, -b e -r

```
./backup_summary.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt -r '.*\.txt$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP

[tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25]$ ./backup_summary.sh -c -b /home/tiago/Desktop/Testes/exclude_list.txt -r '.*\.txt$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
mkdir -p -v "/home/tiago/Desktop/BACKUP"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
eval "bash ./backup_summary.sh" -c -b "/home/tiago/Desktop/Testes/exclude_list.txt" -r '.*\.txt$' "/home/tiago/Desktop/SOURCE/DirA" "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
eval "bash ./backup_summary.sh" -c -b "/home/tiago/Desktop/Testes/exclude_list.txt" -r '.*\.txt$' "/home/tiago/Desktop/SOURCE/DirA/DirC" "/home/tiago/Desktop/BACKUP/DirA/DirC"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirA/DirC"
Arquivo .single file será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Arquivo "file4.exe" será ignorado (flag -r).
Arquivo "files.html" será ignorado (flag -r).
Arquivo "file6.exe" será ignorado (flag -r).
Arquivo ".hidden2" será ignorado (flag -r).
Arquivo "just another space.csv" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
eval "bash ./backup_summary.sh" -c -b "/home/tiago/Desktop/Testes/exclude_list.txt" -r '.*\.txt$' "/home/tiago/Desktop/SOURCE/DirB" "/home/tiago/Desktop/BACKUP/DirB"
mkdir -p -v "/home/tiago/Desktop/BACKUP/DirB"
Arquivo "file7.md" será ignorado (flag -r).
Arquivo "file8.ps" será ignorado (flag -r).
Arquivo "file9.pdf" será ignorado (flag -r).
Arquivo "final space" será ignorado (flag -r).
Arquivo ".hidden3" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

cp -a -v "/home/tiago/Desktop/SOURCE/file1.txt" /home/tiago/Desktop/BACKUP"
Arquivo "file2.log" será ignorado (flag -r).
Arquivo "file3.c" será ignorado (flag -r).
Arquivo ".hidden" será ignorado (flag -r).
Arquivo "literally some space" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Figura 65 – Teste 7



Como esperado, ao utilizarmos a flag -c e as flags -b e -r e, como seus argumentos “exclude_list.txt” e “.*\.txt\$”, respetivamente, o programa simula corretamente a cópia de apenas um ficheiro, sendo esse “file1.txt”, uma vez que termina em ‘.txt’, verificando assim a expressão regular pretendida e o seu caminho/nome não foi encontrado em “exclude_list.txt”.

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Testes Inválidos

Teste 8 – Path de Source Inexistente

```
./backup_summary.sh /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
```

```
[tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25]$ ./backup_summary.sh /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
O caminho "/home/tiago/Desktop/NOTHING" não existe!
While backuping /home/tiago/Desktop/NOTHING: 1 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Figura 66 – Teste 8



Como esperado, ao utilizarmos uma SOURCE inválida (“/home/tiago/Desktop/NOTHING”), o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Para além disso, o programa imprime uma mensagem que indica como correu a execução, neste caso, o número de erros detetados.

Teste 9 – Com Path de Source e Backup iguais

```
./backup_summary.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/SOURCE
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/SOURCE
Os caminhos '/path/to/src' e '/path/to/bkp' não podem ser iguais!
While backuping /home/tiago/Desktop/SOURCE: 1 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)
```

Fim do programa.

Figura 67 – Teste 9



Como esperado, ao utilizarmos um BACKUP igual a SOURCE, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Para além disso, o programa imprime uma mensagem que indica como correu a execução, neste caso, o número de erros detetados.

Teste 10 – Com flag -c e argumentos para -c

```
./backup_summary.sh -c some_argument /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh -c some_argument /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
Parâmetros incorretos!
Esperado: -c -b [tfile] -r [regexp] path/to/src /path/to/bkp
O caminho "some_argument" não existe!
While backuping some_argument: 2 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)
```

Fim do programa.

Figura 68 – Teste 10



Como esperado, ao utilizarmos um argumento (“some_argument”) em conjunto com a flag -c, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro e ainda explica como deve ser feita a introdução de argumentos.

Para além disso, o programa imprime uma mensagem que indica como correu a execução, neste caso, o número de erros detetados.

Teste 11 – Com flag -b e “exclude_list.txt” inválido/não existente

```
./backup_summary.sh -b /home/tiago/Desktop/Testes/non_existent.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh -b /home/tiago/Desktop/Testes/non_existent.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
[tfile] tem de ser um ficheiro válido!
While backuping /home/tiago/Desktop/SOURCE: 1 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)
```

Fim do programa.

Figura 69 – Teste 11



Como esperado, ao utilizarmos um argumento inexistente (“/home/tiago/Desktop/Testes/non_existent.txt”) em conjunto com a flag -b, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Para além disso, o programa imprime uma mensagem que indica como correu a execução, neste caso, o número de erros detetados.

Teste 12 - Com flag -k e Path de Source Inexistente

```
./backup_summary.sh -k /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh -k /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
./backup_summary.sh: illegal option -- k
Parâmetros incorretos!
Esperado: -c -b [tfile] -r [regexp] path/to/src /path/to/bkp
O caminho "/home/tiago/Desktop/NOTHING" não existe!
While backuping /home/tiago/Desktop/NOTHING: 2 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Figura 70 – Teste 12

Como esperado, ao utilizarmos a flag inexistente “-k” e uma SOURCE inválida (“/home/tiago/Desktop/NOTHING”), o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Para além disso, o programa imprime uma mensagem que indica como correu a execução, neste caso, o número de erros detetados.



SOURCE

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE$ ls -a
. .. DirA DirB file1.txt file2.log file3.c .hidden1 'literally some      space'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE$ cd DirA
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA$ ls -a
. .. DirC file4.exe file5.html file6.exe .hidden2 'just      another space.csv'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA$ cd DirC
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA/DirC$ ls -a
. .. '.single file'
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirA/DirC$ cd ../../DirB
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/SOURCE/DirB$ ls -a
. .. file7.md file8.ps file9.pdf 'final      space' .hidden3
```

Figura 71 – Organização interna de SOURCE

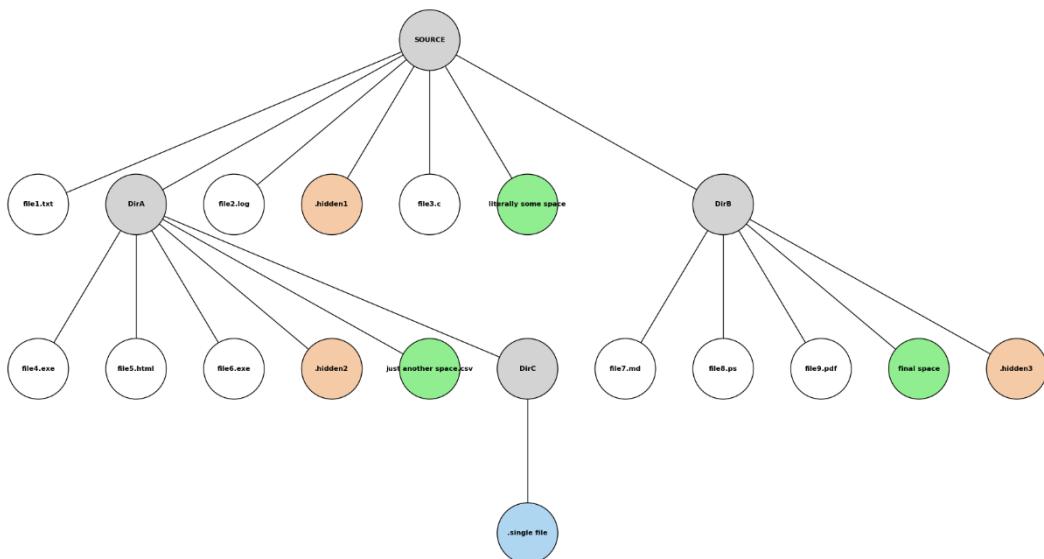


Figura 72 – Grafo de SOURCE

Pastas iguais

Testes Válidos

Teste 1 – Sem flags

```
./backup_check.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_check.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
Fim do programa.
```

Figura 73 – Teste 1



Como esperado, ao não utilizarmos nenhuma *flag*, o programa corre normalmente, apenas apresentando uma mensagem de “Fim de programa”, significando isto que, SOURCE e BACKUP são pastas com o mesmo conteúdo.

Testes Inválidos

Teste 2 – Sem argumentos

```
./backup_check.sh
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_check.sh
Parâmetros incorretos!
Esperado: /path/to/src /path/to/bkp
```

Figura 74 – Teste 2



Como esperado, ao não introduzirmos nenhum argumento, o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Teste 3 – Path de Source Inexistente

```
./backup_check.sh /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_check.sh /home/tiago/Desktop/NOTHING /home/tiago/Desktop/BACKUP
O caminho "/home/tiago/Desktop/NOTHING" não existe!
```

Figura 75 – Teste 3



Como esperado, ao introduzirmos uma SOURCE inexistente (“/home/tiago/Desktop/NOTHING”), o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Teste 4 – Path de Backup Inexistente

```
./backup_check.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/NOTHING
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_check.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/NOTHING  
0 caminho "/home/tiago/Desktop/NOTHING" não existe!
```

Figura 76 – Teste 4



Como esperado, ao introduzirmos um BACKUP inexistente (“/home/tiago/Desktop/NOTHING”), o programa aborta a sua execução e avisa o utilizador sobre o que provocou o erro.

Pastas diferentes (Ficheiros alterados em BACKUP)

Teste 5 - (Criou-se uma pasta cópia de BACKUP "BACKUP_MOD", onde se alterou "file5.html", "final space" e "file2.log")

```
./backup_check.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP_MOD
```

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_check.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP_MOD  
/home/tiago/Desktop/SOURCE/DirA/file5.html /home/tiago/Desktop/BACKUP_MOD/DirA/file5.html differ.  
/home/tiago/Desktop/SOURCE/DirB/final space /home/tiago/Desktop/BACKUP_MOD/DirB/final space differ.  
/home/tiago/Desktop/SOURCE/file2.log /home/tiago/Desktop/BACKUP_MOD/file2.log differ.  
Fim do programa.
```

Figura 77 – Teste 5



Como esperado, ao alterarmos 3 ficheiros (“file5.html”, “final space” e “file2.log”) de BACKUP (equivalente a BACKUP_MOD), o programa irá demonstrar essas diferenças através de 3 mensagens onde afirma que esses 3 ficheiros alterados diferem entre as pastas.

Testes (*a posteriori*)

Estes testes foram efetuados após “BACKUP” ter sido criada pela primeira vez, daí serem *a posteriori*.

backup_files.sh

Teste 1 – Sem flags

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup_files.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BCKUP
Fim do programa.
```

Como esperado, o programa não executa qualquer comando uma vez que BACKUP já se encontra atualizado.

Teste 2 – Remoção e adição de ficheiros em SOURCE (remoção de “just another space.html” e adição de “some new_file.c”)

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---50---2024-25$ ./backup_files.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BCKUP
removed '/home/tiago/Desktop/BCKUP/some new_file.c'
'/home/tiago/Desktop/SOURCE/just      another space.html' -> '/home/tiago/Desktop/BCKUP/just      another space.html'
Fim do programa.
```

Como esperado, após a adição de “some new_file.c” e a remoção de “just another space.csv”, o programa copia e remove os ficheiros corretos, respetivamente.

backup.sh

Teste 1 – Sem flags

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
Fim do programa.
```

Como esperado, o programa não executa qualquer comando uma vez que BACKUP já se encontra atualizado.

Edição de ficheiros em SOURCE (edição de “file5.html”, “file9.pdf” e “literally some space”)

Teste 1 – Sem flags

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
'/home/tiago/Desktop/SOURCE/DirA/file5.html' -> '/home/tiago/Desktop/BACKUP/DirA/file5.html'
'/home/tiago/Desktop/SOURCE/DirB/file9.pdf' -> '/home/tiago/Desktop/BACKUP/DirB/file9.pdf'
'/home/tiago/Desktop/SOURCE/literally some      space' -> '/home/tiago/Desktop/BACKUP/literally some      space'
Fim do programa.
```

Como esperado, após a edição dos 3 ficheiros, o programa copia-os uma vez que são mais atualizados do que os ficheiros que estavam anteriormente em BACKUP.

Teste 2 – Com flag -b

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup.sh -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
'/home/tiago/Desktop/SOURCE/DirA/file5.html' -> '/home/tiago/Desktop/BACKUP/DirA/file5.html'
Arquivo "file9.pdf" será ignorado (flag -b).
'/home/tiago/Desktop/SOURCE/literally some      space' -> '/home/tiago/Desktop/BACKUP/literally some      space'
Fim do programa.
```

Como esperado, após a edição dos 3 ficheiros, o programa apenas não atualiza “file9.pdf” pois o seu caminho/nome encontra-se no ficheiro “exclude_list.txt”.

Teste 3 – Com flag -r

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup.sh -r '.*\.html\$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
'/home/tiago/Desktop/SOURCE/DirA/file5.html' -> '/home/tiago/Desktop/BACKUP/DirA/file5.html'
Arquivo "file9.pdf" será ignorado (flag -r).
Arquivo "literally some      space" será ignorado (flag -r).
Fim do programa.
```

Como esperado, após a edição dos 3 ficheiros, o programa apenas copia “file9.pdf” pois é o único que verifica a expressão regular.

Adição e remoção de ficheiros em SOURCE (adição de “some new_file.c” e remoção de “just another space.csv”)

Teste 1 – Sem flags

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
removed '/home/tiago/Desktop/BACKUP/DirA/just      another space.csv'
'/home/tiago/Desktop/SOURCE/DirA/some new_file.c' -> '/home/tiago/Desktop/BACKUP/DirA/some new_file.c'
Fim do programa.
```

Como esperado, após a adição de “some new_file.c” e a remoção de “just another space.csv”, o programa copia e remove os ficheiros corretos, respetivamente.

backup_summary.sh

Teste 1 – Sem flags

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)
Fim do programa.
```

Como esperado, o programa não executa qualquer comando uma vez que BACKUP já se encontra atualizado.

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Edição de ficheiros em SOURCE (edição de “file5.html”, “file9.pdf” e “literally some space”)

Teste 1 – Sem flags

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/DirA/file5.html' -> '/home/tiago/Desktop/BACKUP/DirA/file5.html'
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 1 Updated; 0 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/DirB/file9.pdf' -> '/home/tiago/Desktop/BACKUP/DirB/file9.pdf'
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 1 Updated; 0 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/literally some    space' -> '/home/tiago/Desktop/BACKUP/literally some    space'
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 1 Updated; 0 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Como esperado, após a edição dos 3 ficheiros, o programa copia-os uma vez que são mais atualizados do que os ficheiros que estavam anteriormente em BACKUP.

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Teste 2 – Com flag -b

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---SO---2024-25$ ./backup_summary.sh -b /home/tiago/Desktop/Testes/exclude_list.txt /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/DirA/file5.html' -> '/home/tiago/Desktop/BACKUP/DirA/file5.html'
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 1 Updated; 0 Copied (0B); 0 Deleted (0B)

Arquivo "file9.pdf" será ignorado (flag -b).
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/literally some    space' -> '/home/tiago/Desktop/BACKUP/literally some    space'
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 1 Updated; 0 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Como esperado, após a edição dos 3 ficheiros, o programa apenas não atualiza “file9.pdf” pois o seu caminho/home encontra-se no ficheiro “exclude_list.txt”.

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Teste 3 – Com flag -r

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_summary.sh -r '.*\.html\$' /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)
'/home/tiago/Desktop/SOURCE/DirA/file5.html' -> '/home/tiago/Desktop/BACKUP/DirA/file5.html'
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 1 Updated; 0 Copied (0B); 0 Deleted (0B)

Arquivo "file9.pdf" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Arquivo "literally some    space" será ignorado (flag -r).
While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Como esperado, após a edição dos 3 ficheiros, o programa apenas copia “file9.pdf” pois é o único que verifica a expressão regular.

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Adição e remoção de ficheiros em SOURCE (adição de “some new_file.c” e remoção de “just another space.csv”)

Teste 1 – Sem flags

```
tiago@tiago-IdeaPad-3-15ALC6:~/Desktop/Trabalho-1---S0---2024-25$ ./backup_summary.sh /home/tiago/Desktop/SOURCE /home/tiago/Desktop/BACKUP
removed '/home/tiago/Desktop/BACKUP/DirA/just    another space.csv'
While backuping /home/tiago/Desktop/SOURCE/DirA/DirC: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

'/home/tiago/Desktop/SOURCE/DirA/some new_file.c' -> '/home/tiago/Desktop/BACKUP/DirA/some new_file.c'
While backuping /home/tiago/Desktop/SOURCE/DirA: 0 Errors; 0 Warnings; 1 Copied (0B); 1 Deleted (0B)

While backuping /home/tiago/Desktop/SOURCE/DirB: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

While backuping /home/tiago/Desktop/SOURCE: 0 Errors; 0 Warnings; 0 Updated; 0 Copied (0B); 0 Deleted (0B)

Fim do programa.
```

Como esperado, após a adição de “some new_file.c” e a remoção de “just another space.csv”, o programa copia e remove os ficheiros corretos, respetivamente.

Para além disso, quando termina de copiar o conteúdo de cada diretoria, o programa imprime uma mensagem de forma a informar como correu a cópia dessa diretoria.

Versões Anteriores

Problemas e Abordagens para resolver

Problema 1

“Como saber se um ficheiro em \$src é ou não mais recente do que em \$bkp (supondo que existe em \$bkp)?”

Primeiramente ponderámos em utilizar o comando *date* para, de alguma forma conseguir comparar as datas de modificação dos dois ficheiros e ver qual deles era mais recente, no entanto, esta abordagem parecia-nos complicada e não viável pelo que, entretanto, fomos confrontados com a possibilidade de usar *-nt* ou *-ot* como forma de comparação. Decidimos optar por *-nt* como forma de comparação.

Esta comparação foi implementada na função *isNewer()*:

```
1  #!/bin/bash
2
3  ↘  function isNewer() {
4      # Caso ainda exista, verificamos se é mais recente ou não
5      if [[ "$1" -nt "$2" ]]; then
6          return 1
7      else
8          return 0
9      fi
10 }
```

Problema 2

“Como detectar a introdução de flags no input?”

A primeira ideia que nos surgiu foi utilizar uma cadeia de *if*'s para contemplar todos os casos possíveis, e foi isso que fizemos, no entanto tornava o código muito extenso e não achámos que fosse uma solução elegante pelo que, após alguma pesquisa, nos deparamos com a função *getopts*.

The diagram illustrates the evolution of a shell script. It starts with a manual approach using nested if statements to handle flags, then transitions to using the *getopts* command, and finally shows how to use *shift* and parameter expansion to manage remaining arguments.

Manual Flag Handling:

```
14 case $# in
15     # Nenhuma Flag ativa
16     2)
17         index=1
18         ;;
19
20     # 1 Flag ativa (c)
21     3)
22         index=2
23         if [[ "$1" == "-c" ]]; then
24             c_flag=1
25             flags="-c"
26         else
27             echo "Parâmetro incorreto. Esperado: '[-c]'"
28             exit 1
29         fi
30         ;;
31
32     # 1 Flag ativa (b/r)
33     4)
34         case $1 in
35             "-b")
36                 if [[ -f $2 ]]; then
37                     b_flag=1
38                     flags="-b $2"
39
40             (...)
```

Using getopts:

```
24     # Processar as opções com getopts
25     while getopts "cb:r:" flag; do
26         case $flag in
27             c)
28                 flags+=" -c"
29                 c_flag=1
30                 ;;
31             b)
32                 b_flag=1
33                 blockedFiles="$OPTARG"
34                 if [[ ! -f "$blockedFiles" ]]; then
35                     echo "[tfile] tem de ser um ficheiro válido!"
36                     exit 1
37                 fi
38                 flags+=" -b $blockedFiles"
39                 ;;
40             r)
41                 r_flag=1
42                 regexpr="$OPTARG"
43                 flags+=" -r \"$regexpr\""
44                 ;;
45             \?)
46                 echo -e "Parâmetros incorretos!\nEsperado: -c -b [tfile] -r [regexpr] /path/to/src /path/to/bkp"
47                 exit 1
48             ;;
49         esac
50     done
```

Using shift and Parameter Expansion:

```
52     # Dá shift das flags e "retira-as" dos argumentos
53     shift $((OPTIND - 1))
54
55     61     # Atribuir os restantes argumentos a src e bkp
56     62     src="$1"
57     63     bkp="$2"
```

The blue arrow points from the manual flag handling section to the *getopts* section, indicating the transition. Brackets on the right side group the code into three main sections: the original manual code, the *getopts* code, and the final code using *shift* and parameter expansion.

Problema 3

“Como copiar ficheiro escondidos e ficheiros com espaço com nomes”

Inicialmente, o nosso programa não copiava esse tipo de ficheiros. Mais tarde, usámos “\$src”{.,} que apenas nos permitia copiar ficheiro escondidos. De forma a encontrar uma solução que nos permitisse copiar tanto ficheiros escondidos como ficheiros cujo nome continha espaços, encontrámos as opções `shopt -s nullglob` e `shopt -s dotglob`, o que nos permitiu solucionar este problema de forma mais simples e eficaz.

```
55  for file_path in "$src"{.,}*; do
56      file_name=$(basename "$file_path")      # remove o prefixo do path e deixa apenas o nomes do ficheiro
57      isNewer "$src$file_name" "$bkp$file_name"
58      if [ $? -eq 1 ]; then
59          # copia-se o ficheiro
60          if [ ${c_flag} -eq 1 ]; then
61              echo "cp -a -v \"$file_path\" $bkp"
62          else
63              cp -a -v "$file_path" $bkp
64          fi
65      fi
66  done
67  echo "Fim do programa."
```



```
60  shopt -s nullglob
61  shopt -s dotglob
62  for path in "$bkp"/*; do
63      name=$(basename "$path")      # remove o prefixo do path e deixa apenas o nome do ficheiro
64
65      # Se o ficheiro já não existir em src, então apagamos de bkp
66      if [ ! -e "$src$name" ]; then
67          if [ ${c_flag} -eq 1 ]; then
68              echo "rm -v \"$path\""
69          else
70              rm -v "$path"
71          fi
72      fi
73  done
74
75  for file_path in "$src"/*; do
76      file_name=$(basename "$file_path")      # remove o prefixo do path e deixa apenas o nome do ficheiro
77
78      # Verifica se o ficheiro é mais recente em src do que em bkp (copia se for mais recente ou se ainda não existir em bkp)
79      isNewer "$src$file_name" "$bkp$file_name"
80      if [ $? -eq 1 ]; then
81          if [ ${c_flag} -eq 1 ]; then
82              echo "cp -a -v \"$file_path\" \"$bkp\""
83          else
84              cp -a -v "$file_path" "$bkp"
85          fi
86      fi
87  done
88  shopt -u nullglob
89  shopt -u dotglob
```

Problema 4

“Como distinguir entre ficheiros atualizado e ficheiros copiados?”

Inicialmente, no programa, *backup_summary.sh*, não sabíamos muito bem como distinguir entre ficheiros atualizados e ficheiros copiados. A solução que mais nos agradou foi criar uma função *checkExistance.sh* que verifica que certo caminho (naturalmente de um ficheiro), existe em \$src antes de ser copiado (retornado \$control=1 se já existir). Como variável auxiliar, criámos \$var (com valor *default*=0), que controla se de facto chegou a haver uma cópia do ficheiro.

No fim do ciclo *for*, verificamos se tinha havia cópia do ficheiro (\$var=1) e depois distinguimos se foi um ficheiro atualizado ou copiado, se \$control=1, então o ficheiro foi atualizado e a variável \$updated era incrementada em 1, caso \$control=0, então o ficheiro foi copiado e a variável \$copied era incrementada em

```
146         isNewer "$file_path" "$bkp/$file_name"
147         # Se ainda existir e for mais recente, então copiamos
148         if [ $? -eq 1 ]; then
149             checkExistance "$bkp/$file_name"      # Utilizado para, no final, sabermos se o ficheiro foi copiado ou atualizado
150             control=$?
151             var=0

233                 # Verificação para ver se o ficheiro não foi ignorado
234                 if [[ $var -eq 1 ]]; then
235                     # Verificação final para atribuir os valores certos a Updated e Copied
236                     if [[ $control -eq 1 ]]; then
237                         ((updated+=1))
238                     else
239                         ((copied+=1))
240                         copied_size=$((copied_size + $(stat -c%s "$file_path")))
241                     fi
242                 fi
```

1.

Bibliografia

(Verificação se a diretoria existia ou não)

<https://stackoverflow.com/questions/59838/how-do-i-check-if-a-directory-exists-or-not-in-a-bash-shell-script>

(Uso do comando cp)

<https://phoenixnap.com/kb/cp-command>

<https://www.geeksforgeeks.org/cp-command-linux-examples/>

(Utilização de expressões regulares (RegEx))

<https://www.geeksforgeeks.org/how-to-use-regular-expressions-regex-on-linux/>

(Remoção de prefixos de paths usando basename)

<https://stackoverflow.com/questions/3294072/get-last dirname-filename-in-a-file-path-argument-in-bash>

(Indirect Expansion)

<https://stackoverflow.com/questions/8515411/what-is-indirect-expansion-what-does-var-mean>

(Leitura de ficheiro line-by-line)

<https://stackoverflow.com/questions/10929453/read-a-file-line-by-line-assigning-the-value-to-a-variable>

(Remoção de ficheiros)

<https://www.earthdatascience.org/courses/intro-to-earth-data-science/open-reproducible-science/bash/bash-commands-to-manage-directories-files/>

(Comparação de datas de ficheiros)

<https://superuser.com/questions/188240/how-to-verify-that-file2-is-newer-than-file1-in-bash>

Conclusão

Este projeto serviu certamente como forma de consolidar os nossos conhecimentos acerca da linguagem *bash* e também sobre os processos do S.O. Linux.

Também aprendemos que programar talvez não seja apenas só programar, temos de pensar como vamos resolver o problema e tentar simplificá-lo ao máximo dividindo-o em sub-problemas cada vez mais pequenos.