universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Trabalho Prático 1
Ferramenta de criação/atualização de cópias de segurança Bash

Relatório – Turma P2

Diogo Duarte Nº 120482

Tomás Hilário Nº 119896

Índice

1. Introdução

Neste trabalho, foi-nos proposto o desenvolvimento de um *script* em *bash* de modo a aplicar os conteúdos lecionados nas aulas práticas e teórico-práticas. O objetivo deste é criar e atualizar uma cópia de segurança de uma diretoria de trabalho em outra diretoria, que pode corresponder a um outro dispositivo (pen usb, disco externo, etc), denominada de backup.

Na primeira fase do trabalho, foi desenvolvidos o script proposto "backup_files.sh" onde era feito o backup de arquivos em um diretório que não contém subdiretórios, com a possibilidade de usar a flag -c para entrar em modo de checking onde não executa os comandos mas exibe-os sem alterar o conteúdo da diretoria backup.

Posteriormente criamos o script "backup.sh" sendo este capaz de realizar o backup de diretórios com subdiretórios, tendo ainda opções adicionais como [-b tfile] e [-r regexpr] explicadas mais tarde neste relatório, criámos ainda o "backup_summary.sh" sendo este script idêntico a "backup.sh" com a adição de incluir como o nome indica um sumário descritivo de cada execução executada, ou seja, o número de erros, warnings, arquivos copiados, arquivos apagados etc.

Por fim criados o último script proposto "backup_check.sh", este itera sobre os arquivos na diretoria de backup e verifica se o seu counteúdo é idêntico ao dos arquivos correspondentes na diretória de trabalho.

2. Desenvolvimento

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

O nosso projeto está dividido essencialmente em 5 partes:

- ✓ Criação de um srcipt para fazer backup de ficheiros (backup_files.sh);
- ✓ Implementação de um script de modo a conseguir fazer backup de diretorios (backup.sh);
- ✓ Alteração do script para mostrar erros e indicações (backup_summary.sh);
- ✓ Verificação dos ficheiros no backup (backup_check.sh);
- ✓ Teste dos dados.

2.1 Backup de ficheiros

Nesta etapa do projeto criámos um script que copia os ficheiros de uma diretória de trabalho para uma diretória de backup, tendo uma opção de checking onde em vez de fazer o backup, faz a impressão dos comandos que seriam usados no terminal.

Este script realiza ainda a atualização dos ficheiros mais recentes que se encontram na diretoria de trabalho para a diretoria de backup através da data de modificação.

Flag c

- Ativa a opção de checking onde o script apresenta na consola todos os comandos , mas não executa nenhum, sendo esta flag opcional.

```
checking=false

while getopts "c" option; do
# itera sobre as opções passadas na linha de comandos e a
rmazena em option
    case $option in
        c)
            checking=true
# modo checking
            ;;
        *)
            echo "Usage: $0 [-c] dir_trabalho dir_backup"
# argumentos inválidos
            exit
            ;;
    esac
done
```

*D*epois removemos os argumentos iterados no loop anterior, ou seja as flags e guardamos os parâmetros de entrada em variaveis, fazemos verificações para saber se a diretoria está vazia, ou se os argumentos passados são invalidos.

```bash
#validação dos argumentos:
shift $((OPTIND - 1))
dir_trabalho="$1"
dir_backup="$2"

if [ $# -ne 2 ] || ! [ -d "$dir_trabalho" ]; then
    echo ">> INVALID ARGUMENTS!!!"
    echo ">> Usage: $0 [-c] dir_trabalho dir_backup"
    exit 1

elif  ! [ -e "$dir_backup" ] || ! [ -d "$dir_backup" ]; then
    echo -e "\n>> WARNING: backup directory \"$dir_backup\" does not exist! Creating it..."
    mkdir "$dir_backup"
fi

if [ -z "$( ls -A $dir_trabalho )" ]; then
    echo -e "\n>> WARNING: source directory \"$dir_trabalho\" is empty!"
    exit 0
fi
```

Enquanto criámos este script encontramos a necessidadede de criar uma função que remove-se os ficheiros que já não se encontravam na diretoria de trabalho mas ainda estavam na diretoria de backup, onde também tem a funcionalidade de utilizar a flag -c e simplesmente apresenta na consola os comandos que seriam executados, esta função recebe 3 argumentos

‘ **rm_olf_files.sh (“$dir_trabalho” “$dir_backup” “$checking”)’** .

```bash
rm_old_files(){
    dir_trabalho="$1"
    dir_backup="$2"
    checking="$3"

    if ! [ -z "$( ls -A $dir_backup )" ]; then             # garante q o dir n está vazio
        for file in "$dir_backup"/{*,.*}; do
            fname="${file##*/}"

            if ! [ -e "$dir_trabalho/$fname" ]; then        # verifica se o ficherio ainda existe n
o dir de trabalho

                if $checking; then
                    echo "rm $dir_backup/$fname"           # printa os comandos estando no modo che
cking
                else
                    rm "$dir_backup/$fname"             # executa os comandos não estando no modo
checking
                    echo -e "\n>> Removed no longer existing file \"$fname\" from \"$dir_backup\"."
                fi
            fi

        done
    fi

}
```

A restante lógica para fazer o backup consiste em verificar se os files que estam na dir_trabalho já se encontram na dir_backup e se esses files são os mais recentes, para isso usamos alguns loops e condições para ver se os ficheiros eram mais recentes, também contenplando a opção de usar flag -c.

```bash
for file in "$dir_trabalho"/{*,.*}; do

    if [[ "$file" == "$dir_trabalho/." || "$file" == "$dir_trabalho/.." ||
"$file" == "$dir_trabalho/.*" || "$file" == "$dir_trabalho/*" ]]; then
# ignorar ".", ".." e ".*"
        continue
    fi
```

```bash
if [[ "$file" -nt "$backed_file" ]]; then
```

Nesta etapa do projeto testamos várias formas de fazer o backup:

- **Pasta de Backup Vazia**

```
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_files.sh ../Teste ../TesteBackup/

>> Copyed "../Teste/SOp2425_aula01.pdf" to "../TesteBackup/".

>> Copyed "../Teste/.texto" to "../TesteBackup/".
```

- **Files em Backup desatualizados**

```
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_files.sh ../Teste ../TesteBackup/

>> File "../Teste/SOp2425_aula01.pdf" doesn't need backing up!

>> File "../Teste/.texto" was successfully updated!
```

- **Diretoria Trabalho vazia**

```
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_files.sh ../Teste ../TesteBackup/

>> WARNING: source directory "../Teste" is empty!
```

- **Files que deixaram de existir na Diretoria Trabalho**

```
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_files.sh ../Teste ../TesteBackup/

>> Removed no longer existing file "../TesteBackup//.texto" from "../TesteBackup/".

>> Copyed "../Teste/SOp2425_aula01.pdf" to "../TesteBackup/".
```

- **Adicionar, atualizar e remover files com flag -c**

```
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_files.sh -c ../Teste ../TesteBackup/
cp -a ../Teste/novo_checking ../TesteBackup/
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_files.sh  ../Teste ../TesteBackup/

>> Copyed "../Teste/novo_checking" to "../TesteBackup/".

>> File "../Teste/SOp2425_aula01.pdf" doesn't need backing up!
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_files.sh -c  ../Teste ../TesteBackup/
cp -a ../Teste/alterado_checking ../TesteBackup/
rm ../TesteBackup//novo_checking
cp -a ../Teste/novo_checking ../TesteBackup/
```

2.2 Backup de ficherios e/ou diretórios

Para esta fase do projeto, criou-se um *script* ***(backup.sh)*** que é capaz de realizar o *backup* de uma diretória que contém ficheiros e sub diretórias, aplicando o mesmo raciocínio que em ***backup_files.sh*** mas permitindo que o *script* se chame a si próprio recursivamente para realizar o *backup* de cada sub diretoria.

- **Flag 'c'**

  Ativa a opção de checking onde o script apresenta na consola todos os comandos , mas não executa nenhum, sendo esta flag opcional.

- **Flag 'b'**

  Permite a indicação de umficheiro de texto que contém uma lista de ficheiros (ou diretorias que não devem ser copiados para a diretoria de backup.

- **Flag 'r'**

  Indica que apenas devem ser copiados os ficheiros que verificam uma expressão regular

Formato da linha de comando:

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

De modo a simplificar o código, no início do *script* foi pré-definido o valor "\w+" para a expressão regular e " " para o nome do ficheiro passado como input e foi declarado o *array* vazio **"dont_update_array"** que irá conter o nome dos ficheiros/diretorias que não devem ser alterados.

Da mesma forma que em ***backup_files.sh*** é utilizado "*getopts*" para identificar as *flags* que foram passadas ao *script* bem como os valores passados. No caso de ter sido passado o nome de um ficheiro é verificado se esse nome realmente representa um ficheiro e se possui permissão de leitura, se o mesmo não se encontra vazio e, caso isto se verifique são lidos para o *array*, os caminhos absolutos dos ficheiros/diretorias que constam nesse ficheiro, caso contrário, é impresso um aviso no terminal e o *array* mantém-se vazio. No caso de ter sido passada uma expressão regular, essa expressão substitui o valor "\w+" posteriormente definido. Se não tiver sido passado nenhum ficheiro ou um ficheiro inválido, o array "dont_update_array" irá manter-se vazio e, da mesma maneira, se não tiver sido passado nenhuma expressão regular o valor da mesma manter-se-á "\w+". Desta maneira, tornar-se-á mais simples verificar se deve ou não ser realizado o backup de um ficheiro e realizar as chamadas recursivas com as *flags* corretas.

```bash
checking=false
tfile=" "                           # valor default para nome do ficheiro para que seja criada uma array vazia no caso de n ter sido dado input tfile
regexpr="\w+"                       # expressao regular que aceita todos os nomes de ficheiros, garantindo que se não for dada uma expressão regular, todos os ficheiros são atualizados
declare -a dont_update              # declarar array vazia que servirá para armazenar nomes de ficheiros a não atualizar no caso de ser passado algum pelo input tfile

while getopts "cb:r:" option; do        # itera sobre as opções passadas na linha de comandos e armazena em option
    case $option in
        c)
            checking=true               # ativa modo checking
            ;;
        b)
            tfile="$OPTARG"                                 # guarda em tfile o nome do ficheiro passado
            if [ -f "$tfile" ] && ! [ -z "$tfile" ]; then       # garante que é um ficheiro e não está vazio antes de iterar pelos ficheiros nele escrito e guardar na array
                index=0

                while read -r line; do
                    dont_update[$index]=$(realpath "$line")                 # coloca no array "dont_update" path absoluto dos ficheiros que não serão atualizados no backup
                    index=$(($index+1))
                done < "$tfile"

            elif [ "$tfile" == " " ]; then
                continue
                                                                            # imprimir warning caso tfile n exiista
            else
                echo -e "\n>> WARNING: tfile \"$tfile\" does not exist!"

            fi
            ;;
        r)
            regexpr="$OPTARG"                               # guarda em regexpr a expressão regular passada
            ;;

        *)
            echo "Usage: $0 [-c] [-b tfile] [-r regexpr] dir_trabalho dir_backup"   # output caso haja algo inválido
            exit
            ;;
    esac
done
```

Depois de terem sido lidos os valores introduzidos pelo utilizador e os mesmos terem sido verificados, é chamada a função ***rm_old_files2.sh*** (ver funcionamento) para que sejam removidos quaisquer ficheiros ou diretórias que já não existam na diretória de trabalho atual.

Posto isto, podemos então dar início à iteração sobre todos os itens da diretória de trabalho, incluindo aqueles que estão escondidos. Esta parte do código foi dividido em duas partes: uma para caso o item iterado seja um ficheiro e outra caso seja uma diretória. No caso do item se tratar de um ficheiro, é realizado o mesmo processo que em ***backup_files*** com uma condição extra que garante que o ficheiro não consta na lista de ficheiros a não atualizar, com recurso à função ***in_array*** (ver funcionamento) e que verifica a expressão regular. Como tínhamos concluído acima graças a forma como foi definido o *array* e a expressão regular, essa verificação será concluída em apenas uma linha.

```bash
if [ -f "$item" ]; then                                 # caso item seja um ficheiro
        file="$item"
        fname="${file##*/}"                                 # tirar nome do ficheiro
        absolute_path=$(realpath "$file")                   # obter path absoluto do ficherio para poder verficar se consta na array "dont_update"
        in_array "$absolute_path" "${dont_update[@]}"       # verificar se esse ficherio consta na list de ficherios a não atualizar
        ret_val=$?                                          # valor de retorno da função (1: está no array; 0: não está no array)

        if [ "$ret_val" -eq 0 ] && [[ "$file" =~ $regexpr ]]; then      # garantir que ficherio n está no array e valida a expressão regular que não sendo passada nenhuma será "\w+" e aceitará qq ficheiro

```

No caso em que o item sobre o qual estamos a iterar se trata de uma diretoria, o *script* cria esse diretório na pasta de backup se ainda não existir e não estiver no modo de *checking* e chama-se recursivamente sobre o mesmo para que possa ser feita a cópia dessa diretoria. No que toca às chamadas recursivas, graças à simplificação explicada acima, apenas precisamos de verificar se nos encontramos ou não no modo de *checking* e chamar com ou sem a *flag* **“-c”**, respetivamente. As *flags* **“-b”** e **“-r”** são sempre colocadas nas chamadas recursiva, pois não afetam negativamente o processo de backup.

Nesta etapa do projeto, entre outras, testamos várias formas de fazer o backup:

- **Realizar backup sem passar nenhuma *flag***

```
tsh19@pctomas:~/Desktop/SO/P/Trabalho1_SO$ ./backup.sh ../src ../backup

>> WARNING: backup directory "../backup" does not exist! Creating it...

>> Copyed "../src/file1" to "../backup".

>> Created directory "../backup/subdir1" in "../backup"

>> Copyed "../src/subdir1/subfile1.txt" to "../backup/subdir1".

>> Created directory "../backup/subdir2" in "../backup"

>> Copyed "../src/subdir2/subfile2.txt" to "../backup/subdir2".
```

- **Realizar backup com *flag* r**

```
tsh19@pctomas:~/Desktop/SO/P/Trabalho1_SO$ ./backup.sh -r "^.*\.txt$" ../src ../backup

>> WARNING: backup directory "../backup" does not exist! Creating it...

>> File "../src/file1" will not be updated due to user input (tfile or regex)!

>> Created directory "../backup/subdir1" in "../backup"

>> Copyed "../src/subdir1/subfile1.txt" to "../backup/subdir1".

>> Created directory "../backup/subdir2" in "../backup"

>> Copyed "../src/subdir2/subfile2.txt" to "../backup/subdir2".
```

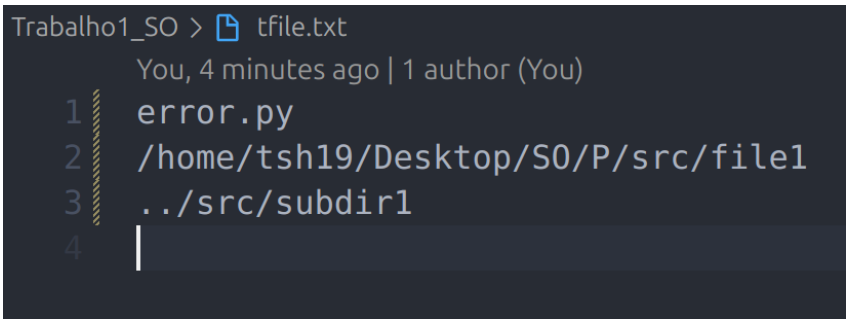
- **Realizar backup com *flag* b**

```
tsh19@pctomas:~/Desktop/SO/P/Trabalho1_SO$ ./backup.sh -b "tfile.txt" ../src ../backup

>> WARNING: backup directory "../backup" does not exist! Creating it...

>> File "../src/file1" will not be updated due to user input (tfile or regex)!

>> Directory "../src/subdir1" will not be updated due to user input (tfile)!

>> Created directory "../backup/subdir2" in "../backup"

>> Copyed "../src/subdir2/subfile2.txt" to "../backup/subdir2".
```

- **Realizar backup com *flag* r e b simultaneamente**

```
tsh19@pctomas:~/Desktop/SO/P/Trabalho1_SO$ ./backup.sh -b "tfile.txt" -r "^.*\.txt$" ../src ../backup
>> WARNING: backup directory "../backup" does not exist! Creating it...
>> File "../src/file1" will not be updated due to user input (tfile or regex)!
>> Directory "../src/subdir1" will not be updated due to user input (tfile)!
>> Created directory "../backup/subdir2" in "../backup"
>> Copyed "../src/subdir2/subfile2.txt" to "../backup/subdir2".
```

- **Conteúdo do "tfile":**



2.3 Indicações de Erros e Warnings

Nesta fase, criamos o *script* **backup_summary.sh** que funciona de igual modo ao *script* anterior mas possui a capacidade de imprimir para cada diretoria uma mensagem que contém o número de ficheiros copiados, atualizados, não alterados e apagados, bem como o número de bytes que cada um destes ocupava. Isto foi possível graças à inicialização de algumas variáveis que serão incrementadas à medida que se realizam ações sobre ficheiros e/ou diretorias em cada diretoria de trabalho.

```
if ! $checking; then
    echo -e "\n>> While backuping $dir_trabalho: $errors Errors;
    $warnings Warnings; $updated Updated; $copied Copied ($bytes_copied B);
    $untouched Untouched ($bytes_untouched B);  $deleted Deleted ($bytes_deleted B)"
fi
```

```
errors=0
warnings=0
updated=0
copied=0
deleted=0
untouched=0
bytes_copied=0
bytes_deleted=0
bytes_untouched=0
```

Nesta etapa do projeto, entre outras, testamos várias formas de fazer o backup:

- **Realizar backup sem passar nenhuma *flag***

```
tsh19@pctomas:~/Desktop/SO/P/Trabalho1_SO$ ./backup_summary.sh ../src ../backup
>> WARNING: backup directory "../backup" does not exist! Creating it...
>> Copyed "../src/file1" to "../backup".
>> Created directory "../backup/subdir1" in "../backup"
>> Copyed "../src/subdir1/subfile1.txt" to "../backup/subdir1".
>> While backuping ../src/subdir1: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (14 B); 0 Untouched (0 B); 0 Deleted (0 B)
>> Created directory "../backup/subdir2" in "../backup"
>> Copyed "../src/subdir2/subfile2.txt" to "../backup/subdir2".
>> While backuping ../src/subdir2: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (9 B); 0 Untouched (0 B); 0 Deleted (0 B)
>> While backuping ../src: 0 Errors; 1 Warnings; 0 Updated; 1 Copied (28 B); 0 Untouched (0 B); 0 Deleted (0 B)
```

- **Realizar backup com *flag* r**

```
tsh19@pctomas:~/Desktop/SO/P/Trabalho1_SO$ ./backup_summary.sh -r "^.*\.txt$" ../src ../backup
>> WARNING: backup directory "../backup" does not exist! Creating it...
>> File "../src/file1" will not be updated due to user input (tfile or regex)!
>> Created directory "../backup/subdir1" in "../backup"
>> Copyed "../src/subdir1/subfile1.txt" to "../backup/subdir1".
>> While backuping ../src/subdir1: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (14 B); 0 Untouched (0 B); 0 Deleted (0 B)
>> Created directory "../backup/subdir2" in "../backup"
>> Copyed "../src/subdir2/subfile2.txt" to "../backup/subdir2".
>> While backuping ../src/subdir2: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (9 B); 0 Untouched (0 B); 0 Deleted (0 B)
>> While backuping ../src: 0 Errors; 1 Warnings; 0 Updated; 0 Copied (0 B); 1 Untouched (28 B); 0 Deleted (0 B)
```

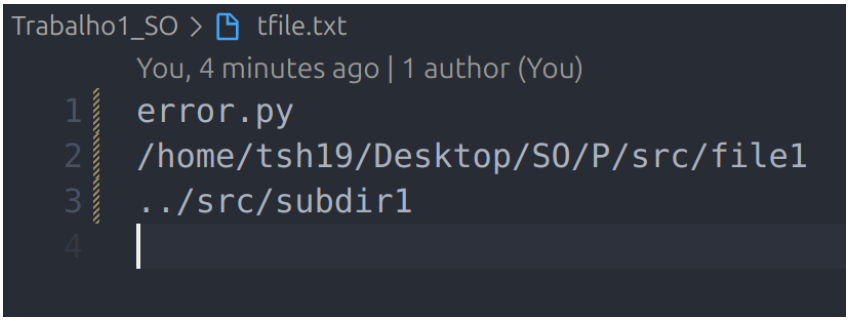
- **Realizar backup com *flag* b**

```
tsh19@pctomas:~/Desktop/SO/P/Trabalho1_SO$ ./backup_summary.sh -b "tfile.txt" ../src ../backup
>> WARNING: backup directory "../backup" does not exist! Creating it...
>> File "../src/file1" will not be updated due to user input (tfile or regex)!
>> Directory "../src/subdir1" will not be updated due to user input (tfile)!
>> Created directory "../backup/subdir2" in "../backup"
>> Copyed "../src/subdir2/subfile2.txt" to "../backup/subdir2".
>> While backuping ../src/subdir2: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (9 B); 0 Untouched (0 B); 0 Deleted (0 B)
>> While backuping ../src: 0 Errors; 1 Warnings; 0 Updated; 0 Copied (0 B); 2 Untouched (42 B); 0 Deleted (0 B)
```

- **Realizar backup com *flag* r e b simultaneamente**

```
tsh19@pctomas:~/Desktop/SO/P/Trabalho1_SO$ ./backup_summary.sh -b "tfile.txt" -r "^.*\.txt$" ../src ../backup

>> WARNING: backup directory "../backup" does not exist! Creating it...

>> File "../src/file1" will not be updated due to user input (tfile or regex)!

>> Directory "../src/subdir1" will not be updated due to user input (tfile)!

>> Created directory "../backup/subdir2" in "../backup"

>> Copyed "../src/subdir2/subfile2.txt" to "../backup/subdir2".

>> While backuping ../src/subdir2: 0 Errors; 0 Warnings; 0 Updated; 1 Copied (9 B); 0 Untouched (0 B); 0 Deleted (0 B)

>> While backuping ../src: 0 Errors; 1 Warnings; 0 Updated; 0 Copied (0 B); 2 Untouched (42 B); 0 Deleted (0 B)
```

- **Conteúdo do "tfile":**



Foram também realizados testes que permitiram verificar a veracidade dos valores impressos no terminal, por exemplo adicionando ou removendo ficheiros com um determinado número de bytes que depois era comparado com os valores obtidos pelo *script*.

2.4 Verificação do backup

Nesta etapa do projeto, criamos um script para verificar se o conteúdo dos arquivos na diretoria de backup é idêntico ao conteúdo correspondente na diretoria de trabalho. Para isso, implementou-se uma função recursiva, ***"checkrec"***, que percorre todos os arquivos e diretórios da **diretoria backup**.

1. **Função Recursiva** checkrec:
    - A função checkrec percorre cada item da diretoria de backup, verificando se existe um item correspondente na diretoria de trabalho, tendo uma abordagem recursiva quando esse item é uma diretória.

- **Arquivos**: Para cada arquivo encontrado, a função compara o conteúdo através de checksums MD5. Se os checksums não coincidirem, uma mensagem de erro é gerada, indicando que os arquivos diferem.

```
if [ -f "$item2" ];then
            item1="$dir1/$(basename "$item2")"

            if [ -f "$item1" ];then

                if [ "$(md5sum "$item2" | awk '{ print $1 }')" != "$(md5sum "$item1" | awk '{ print $1 }')" ]; then
                    echo -e "\n>> \""$item2"\" \""$item1"\" differ."
                    differ="true"
                fi

            else
                echo -e "\n>> \""$item2"\" not in \""$dir_trabalho"\""
                differ="true"
            fi
```

- **Diretórios**: Caso o item seja uma subdiretoria, a função se chama recursivamente para verificar todos os elementos internos, garantindo uma análise completa de cada nível de subdiretórios.

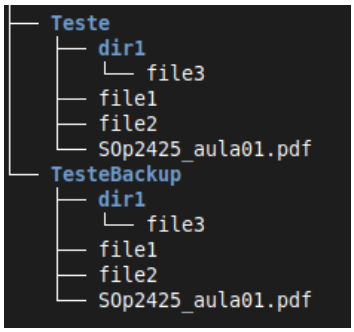
```
elif [ -d "$item2" ]  ; then

            item1="$dir1/$(basename "$item2")"

            if [ -d "$item1" ];then
                checkrec "$item1" "$item2"

            else
                echo -2 "\n>> \""$item2"\" not in \""$dir_trabalho"\""
                differ="true"
            fi
        fi
```

2. **Mensagens de Erro e Verificação Final**:
   - Se um item está ausente ou difere no conteúdo, o script imprime uma mensagem correspondente.
   - Após a execução da função, se nenhuma diferença for detectada, o script exibe ”>> Both directories are equal!!! ” indicando que o backup é fiel à diretoria de trabalho.

Nesta ultima etapa realizamos os seguintes testes

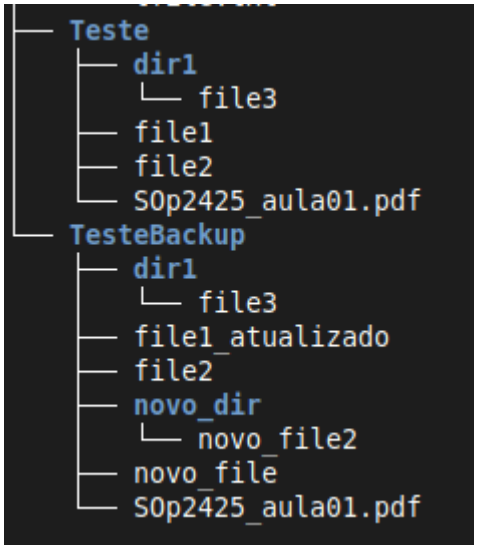
- **Execução do script com todos os itens do dir_backup presentes e atualizados em dir_trabalho**

```
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_check.sh ../Teste ../TesteBackup/
>> All correspondent items in dir_backup are in dir_trabalho !!!
```



- **Execução do script com itens desatualizados ou inexistentes**

```
diogo@diogo:~/Documentos/SO/Projeto1_S0$ ./backup_check.sh ../Teste ../TesteBackup/

>> "../TesteBackup//file1_atualizado" not in "../Teste"

>> "../TesteBackup//novo_dir" not in "../Teste"

>> "../TesteBackup//novo_file" not in "../Teste"
```



2.5 Ficheiros/funções adicionais desenvolvidas

"rm_old_files.sh" (1ª versão)

Usado numa primeira instância do projeto em ***backup_files.sh*** sendo a sua função iterar sobre todos os ficheiros na diretoria de backup (com exceção dos casos em que a diretoria se encontra vazia) e verificar se os mesmos existem na diretoria de trabalho. No caso em que isto não se verifica, estes ficheiros são apagados e é impressa no terminal uma mensagem para o utilizador. A função não deixa de ter em consideração se o *script* se encontra ou não no modo de *checking*, agindo em função disso.

"rm_old_files2.sh" (2ª versão)

Sendo esta função uma melhoria de ***rm_old_files*** a função ***rm_old_files2*** baseia-se nos mesmos princípios da função anterior, mas aceita a possibilidade dos itens da diretoria de backup serem diretórios, tratando os dois casos de forma diferente. Esta função possui a capacidade de guardar o número de ficheiros e/ou diretorias apagados, bem como o número de bytes que os mesmos ocupavam, retornando esses valores com o auxílio de um ficheiro temporário. Estes valores podem não ser retornados omitindo o 4º argumento que representa o *path* do ficheiro temporário. Dessa maneira a função pode ser utilizada tanto no **backup.sh** como no **backup_summary.sh.**

"in_array.sh"

Ficheiro que implementa a função ***in_array*** que verifica se um elemento pertence a um *array* passado como argumento. Neste caso pretendemos verificar se o nome de um ficheiro/diretório se encontra no *array* que contém a lista de ficheiros/diretórios que não devem ser alterados. Retorna "1" se o nome do ficheiro/diretório se encontra no *array* e "0" se o mesmo não se encontra no *array*.

3. Fontes

Para este trabalho, utilizamos as seguintes fontes:

- https://medium.com/@wujido20/handling-flags-in-bash-scripts-4b06b4d0ed04
- https://linuxize.com/post/bash-functions/
- https://www.uptimia.com/questions/how-to-compare-file-dates-in-bash
- https://superuser.com/questions/352289/bash-scripting-test-for-empty-directory
- https://www.masteringunixshell.net/qa40/bash-how-to-pass-array-to-function.html

4. Conclusão

Com este trabalho aprendemos como utilizar a linguagem "*bash*" para realizar operações sobre diretorias que possam ou não ser constituídas por ficheiros/subdiretorias escondidas. Aprendemos a ler ficheiros e *arrays*, utilizar expressões regulares, bem como alguns métodos inerentes ao *bash* como "*getopts*", *wildcards* para iterar sobre ficheiros escondidos, "*shift*" para retirarmos os argumentos que nos interessavam, entre outros… De um modo geral, consideramos este projeto muito enriquecedor e útil para a nossa aprendizagem sobre a cadeira de sistemas operativos.