



PRACTICA 1

FONAMENTS DE SISTEMES OPERATIUS



MARÇ 2024

ALFONSO SANCHEZ FERRER
EDUARD VERICAT BATALLA

ÍNDEX

1. DECISIONS DE DISENY.	2
1.1. DECISIONS DE DISENY SCRIPT COPIA.	2
1.2. DECISIONS DE DISENY SCRIPT RECUPERAR.	3
2. CODIS FETS.	5
2.1. COPIA.sh	5
2.2. RECUPERA.sh	7
3. JOCS DE PROVES.	9
3.1. JOCS DE PROVES Copia.sh	9
3.2. JOCS DE PROVES Recupera.sh	11
4. EXECUCIO CRONTAB TASCA COPIA.SH	14

1. DECISIONS DE DISENY.

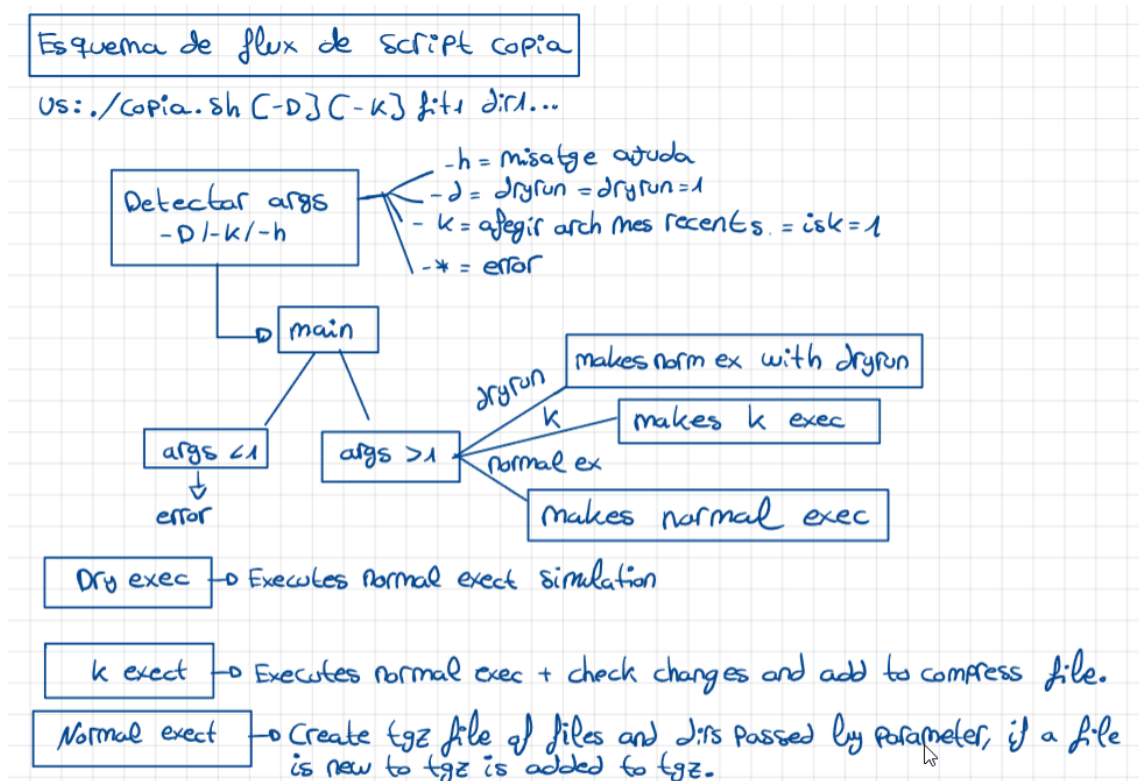
1.1. DECISIONS DE DISENY SCRIPT COPIA.

Per a aquest script hem decidit tractar tot de la manera mes estructurada possible. Es a dir, aquest script es podria arribar a fer tot seguit sense fer us de cap funció, però creiem que es molt bo fer estructura i funcions per tal de fer un script molt mes fàcil d'escalar.

Aquesta estratègia ens permet poder detectar errors molt ràpidament i localitzar-los de forma molt mes senzilla que si fos un codi implementat sense cap mena d'estructura.

Cal dir que aquesta idea, també ha portat una petita dificultat a l'hora de fer la programació del script. Però mes be això ha sigut per estar aprenent encara el llenguatge de Shell.

Així doncs tenim el codi segmentat segons el següent esquema:



També hem decidit experimentar una mica amb bash, i hem descobert el us de colors en els echo. Permetint outputs molt mes interessants. Exemple:

```
alfonso@alfonso-1-2:~/Documentos/SCRIPT COPIA$ ./copia.sh -D pepe.txt TESTING/1 TESTING/carpeta1
Afeint > (f)pepe.txt a llista fitxers
Afeint > (f)TESTING/1 a llista fitxers
Afeint > (d)TESTING/carpeta1 a llista directoris.
Se ha afegit la llista generada al archiu Copia_tgz!!
```

També hem tingut molt en compte les rutes relatives i absolutes, per això, el script sempre podrà copiar totes les rutes que se li pasen per paràmetre. El únic error que existeix, es que les carpetes i fitxers no poden tindre espais.

També he afegit una variable que es una ruta on es vol guardar el .tgz. Aquesta variable es útil per definir on volem guardar la copia de seguretat i controlar el fet de que el script utilitza el path relatiu segons d'on es crida.

1.2. DECISIONS DE DISENY SCRIPT RECUPERAR.

L'script `recuperar.sh` proporciona un conjunt de funcions per facilitar la comprensió i la manipulació correctament dels fitxers directoris.

En primer lloc, la funció ***existeix*** verifica l'existència d'un fitxer o directori donat, retorna un valor booleà en funció del resultat.

La funció ***recuperar*** és el nucli del script, aquesta s'encarrega d'extreure els arxius d'un fitxer comprimit i copiar-los al directori de destinació especificat. També utilitza una estructura de control de flux per controlar la possibilitat de simular (mode Dry Run) o executar-la realment. Aquesta modularitat permet flexibilitat en l'execució del script, ja que l'usuari pot provar l'acció sense afectar els arxius reals.

Un **punt clau** de disseny és l'ús de variables significatives com ***directori temporal*** i ***directori desti.***

El script també fa ús d'estructures de bucle `for` i `while` per recórrer arxius i directoris de la manera més eficient possible.

Esquema del codi:

Entrada:

- Un fitxer comprimit en format .tgz o .tar.gz
- Una llista de noms d'arxius o directoris a recuperar
- Opcionalment, l'indicador "-D" per executar en mode de simulació (Dry Run)

Sortida:

- Els arxius o directoris especificats es copien del fitxer comprimit a la carpeta de destinació.
- Si s'executa en mode Dry Run, es mostra una simulació de les accions que es realitzarien.

Funcions:

- ***existeix***: Verifica si un fitxer o directori existeix.
- ***descomprimir_temp***: Descomprimeix el fitxer comprimit en un directori temporal.
- ***recuperar***: Recorre el contingut del directori temporal i copia els arxius o directoris que coincideixin amb els noms especificats a la llista.

Detalls addicionals:

- La funció ***recuperar*** utilitza un bucle ***while*** per recórrer el contingut del directori temporal.

- S'utilitza la variable **dryRun** per determinar si s'ha de realitzar la còpia d'arxius o només mostrar una simulació.
- S'utilitzen diferents colors de text per a resaltar els missatges d'error.

2. CODIS FETS.

2.1. COPIA.sh

Fet en imatges ja que si copiem al ser línies tan llargues queda el codi malament i molt difícil de llegir.

```
1#!/bin/bash
2
3name="Copia.tgz" #Name of the .tar file
4tempDir="temp" #Temporary dir for testing and checking dates
5fullpath="/home/milax/COPIA"
6
7# Use message
8use="Us: $(basename "$0") [-h] [-K]/[-D] [arxiu/directoris]
9
10On:
11-h Mostra aquest missatge
12-K Comprova si fitxer esta al comprimit, si esta, afegeix el nou amb la data d'ahui.
13-D DryRun, no farà res, sols mostra per pantalla les execucions que faria el script.
14
15Examples:
16
17> Afegir arxius mes nous que les de la copia
18$(tput setaf 1) $(basename "$0")$(tput sgr0) $(tput setaf 5)-K $(tput sgr0)$(tput setaf 6)arxiu_3 arxiu_2 dir_1 $(tput sgr0)
19
20> Mostrar tot el que farà el script sense modificar res
21$(tput setaf 1) $(basename "$0")$(tput sgr0) $(tput setaf 5)-D $(tput sgr0)$(tput setaf 6)dir_1 arxiu_4 $(tput sgr0)
22
23> Generar una copia de seguretat si no existeix $kap o afegir a la copia actual si existeix
24$(tput setaf 1) $(basename "$0")$(tput sgr0)$(tput setaf 6) arxiu_3 arxiu_4 arxiu_5 dir_3 $(tput sgr0)
25
26"
27
28# Functions developed
29
30#Compress files checking the args, if is -D only show how is gonna do it. If is -K checks and changes name to the recent date.
31function compressFiles() {
32    if [ ! -e "$fullpath/$name" ]; then #If the copy doesnt exists, create it
33        if [ $dryRun -eq 1 ]; then
34            echo "$(tput setaf 1)Se ha creat el archiu $name amb la llista generada!!$(tput sgr0)"
35        else
36            echo "creat tgz"
37            tar cfz $fullpath/$name $dirList $fileList
38        fi
39    else
40        echo "Voy donde no tengo que ir"
41        if [ $dryRun -eq 1 ]; then
42            echo "$(tput setaf 1)Se ha afegit la llista generada al archiu $name!!$(tput sgr0)"
43        elif [ $isK -eq 1 ]; then
44            mkdir $fullpath/$tempDir
45            tar xzf $fullpath/$name -C $fullpath/$tempDir
46            for file in $fileList; do
47                fileWithDate=$(date +%Y%m%d)
48                mod_time_temp=$(stat -c %Y "$fullpath/$tempDir/$file")
49                mod_time_original=$(stat -c %Y "$file")
50                if [ "$mod_time_original" -gt "$mod_time_temp" ]; then
51                    cp $file $fullpath/$tempDir/$fileWithDate
52                fi
53            done
54            # We compress files again
55            rm -f $fullpath/$name
56            cd $fullpath/$tempDir
57            tar cfz ../$name * #verify
58            cd ..
59            rm -R $fullpath/$tempDir
60        else
61            #Uncompress files, make changes and compress again
62            mkdir $fullpath/$tempDir
63            tar xzf $fullpath/$name -C $fullpath/$tempDir
64            for dir in $dirList; do
65                cp -r $dir $fullpath/$tempDir/$dir
66            done
67            for file in $fileList; do
68                cp $file $fullpath/$tempDir/$file
69            done
70            rm -f $name
71            cd $fullpath/$tempDir
72            pwd
73            tar cfz ../$name *
74            cd ..
75            rm -R $fullpath/$tempDir
76        fi
77    fi
78}
79
80#Generate a list of files and dirs passed by args.
```

```

#Generate a list of files and dirs passed by args.
function generateList() {
    dirList=""
    fileList=""
    for input in $argsList; do
        if [ -d "$input" ]; then
            if [ $dryRun -eq 1 ]; then
                echo "${tput setaf 1}$fegint$(tput sgr0) > $(tput setaf 7)(d)$(tput sgr0)$(tput setaf 5)$input$(tput sgr0) a $(tput setaf 6)llista directoris.$(tput sgr0)"
            else
                dirList="$dirList $input"
            fi
        else
            if [ $dryRun -eq 1 ]; then
                echo "${tput setaf 1}$fegint$(tput sgr0) > $(tput setaf 7)(f)$(tput sgr0)$(tput setaf 5)$input$(tput sgr0) a $(tput setaf 6)llista fitxers.$(tput sgr0)"
            else
                fileList="$fileList $input"
            fi
        fi
    done
}

```

```

function main() {
    if [ $numberArgs -ge 1 ]; then
        if [ ! -e $fullpath ]; then
            mkdir $fullpath
            generateList
            compressFiles
        else
            generateList
            compressFiles
        fi
        exit 0
    elif [ $numberArgs -eq 0 ]; then
        echo "$use"
        exit 1
    fi
}

```

```

#Boolean vars for checking args
dryRun=0
isK=0

```

```

#Check the args, for -h -K -D options
while getopts 'hKD' option; do

```

```

    case "$option" in
        h) echo "$use"
            exit
            ;;
        K) isK=1
            shift
            numberArgs=$#
            argsList=$@
            main
            exit
            ;;
        D) dryRun=1
            shift
            numberArgs=$#
            argsList=$@
            main
            exit
            ;;
        \?) #A beauty error message
            echo "${tput setaf 1}*****$(tput sgr0)"
            echo "${tput setaf 2}          /\ /|          $(tput sgr0)"
            echo "${tput setaf 3}        |||| |          $(tput sgr0)"
            echo "${tput setaf 4}        \ | \          $(tput sgr0)"
            echo "${tput setaf 5}        /  ()()   iii ERROR !!!  $(tput sgr0)"
            echo "${tput setaf 6}      / - - \ =>*<=          $(tput sgr0)"
            echo "${tput setaf 7}    /|      \ /|          $(tput sgr0)"
            echo "${tput setaf 8}    \|      / \|          $(tput sgr0)"
            echo "${tput setaf 9}    \_____/ \__\          $(tput sgr0)"
            echo "${tput setaf 1}*****$(tput sgr0)"
            echo ""
            echo "$use" >&2
            exit 1
            ;;
    esac
done

```

```

#if no errors or arguments go to normal execution

```

```

numberArgs=$#
argsList=$@
main

```

2.2. RECUPERA.sh.

```
1  #!/bin/bash
2
3  # Funció per verificar si existeix el fitxer o el directori
4  function existeix() {
5      if [ -e "$1" ]; then
6          return 0
7      else
8          return 1
9      fi
10 }
11
12 # Funció per a descomprimir temporalment el directori
13 function descomprimir_temp() {
14     directori_temporal=$(mktemp -d)
15     tar -xzf "$1" -C "$directori_temporal" || { echo "ERROR al descomprimir l'arxiu."; exit 1; }
16 }
17
```

```
function recuperar() {
    # Verifiquem si estem en mode Dry Run o creem la carpeta destí
    if [ "$dryRun" = false ]; then
        mkdir -p "$directori_desti" || { echo "Error en crear la carpeta destí."; exit 1; }
    else
        echo "Creant carpeta DESTÍ amb el nom $directori_desti ..."
    fi

    # Descomprim l'arxiu al directori temporal
    descomprimir_temp "$1"

    # Recorrem tots els arxius i carpetes del directori temporal
    find "$directori_temporal" -mindepth 1 -print0 | while IFS= read -r -d '$\0' ruta_final; do
        nombre_archivo=$(basename "$ruta_final")

        # Comprovem si coincideix amb algun paràmetre, excepte l'últim
        for ((i = 1; i < $#; i++)); do
            if [ "$nombre_archivo" == "${!i}" ]; then
                # Si coincideix i és un arxiu, el copiem al directori destí
                if [ -f "$ruta_final" ]; then
                    if [ "$ruta_final" != "${directori_temporal}/${!i}" ]; then
                        if [ "$dryRun" = true ]; then
                            echo "Copiant $ruta_final a $directori_desti..."
                        else
                            cp "$ruta_final" "$directori_desti" || { echo "Error en copiar $ruta_final a $directori_desti."; exit 1; }
                        fi
                    fi
                fi
            fi
        done

        # Busquem arxius amb el mateix nom base però amb qualsevol extensió després del punt
        nombre_base="${!i%.*}"
        find "$directori_temporal" -maxdepth 1 -type f -name "${nombre_base}.*" ! -name "${!i}" -print0 | while IFS= read -r -d '$\0' archivo_ext; do
            if [ "$archivo_ext" != "$ruta_final" ]; then
                if [ "$dryRun" = true ]; then
                    echo "Copiant $archivo_ext a $directori_desti..."
                else
                    cp "$archivo_ext" "$directori_desti" || { echo "Error en copiar $archivo_ext a $directori_desti."; exit 1; }
                fi
            fi
        done
    done

    # Verifiquem si és un directori que coincideix amb un paràmetre
    if [ -d "$ruta_final" ]; then
        for ((i = 1; i < $#; i++)); do
            if [ "${ruta_final##*/}" == "${!i}" ]; then
                if [ "$dryRun" = true ]; then
                    echo "Copiant directori $ruta_final a $directori_desti..."
                else
                    cp -r "$ruta_final" "$directori_desti/${!i}" || { echo "Error en copiar $ruta_final a $directori_desti."; exit 1; }
                fi
            fi
        done
    fi

    # Si no estem en mode Dry Run, eliminem el directori temporal
    if [ "$dryRun" = false ]; then
        rm -rf "$directori_temporal"
    fi
}

```



```

82 # Inicialitzem la variable que ens indica si recuperem informació o no
83 dryRun=false
84 directori_temporal=""
85 directori_desti="${!#}"
86
87 #COMENÇAMENT DEL MAIN
88
89 if [ $# -lt 3 ]; then
90     echo "ERROR: Paràmetres insuficients !!!"
91     echo "$(tput setaf 1)*****$(tput sgr0)"
92     echo ""
93     echo "$(tput setaf 2)          /\ /\          $(tput sgr0)"
94     echo "$(tput setaf 3)          |||| |          $(tput sgr0)"
95     echo "$(tput setaf 4)          \ | \          $(tput sgr0)"
96     echo "$(tput setaf 5)          _ _ / ()() ERROR      $(tput sgr0)"
97     echo "$(tput setaf 6)          /   \  =>*<=          $(tput sgr0)"
98     echo "$(tput setaf 7)          /|      \  /|          $(tput sgr0)"
99     echo "$(tput setaf 8)          \|      /_ | |          $(tput sgr0)"
100     echo "$(tput setaf 9)          \____) \_)          $(tput sgr0)"
101     echo "$(tput setaf 1)*****$(tput sgr0)"
102     mostrar_menu
103     exit 1;
104 else
105
106     # Comprovem si s'ha introduït un "-D" com a primer parametre
107     if [ "$1" == "-D" ]; then
108         dryRun=true
109         echo "Executant en mode de simulació (Dry Run)..."
110         # Eliminem el primer parametre de la llista (en aquest cas la D)
111         shift
112     fi
113
114     # Comprovació si el "fitxer.tgz" existeix
115     if existeix "$1"; then
116
117         # Descomprimim l'arxiu
118         descomprimir_temp "$1"
119
120         recuperar "$@"
121
122     else
123         echo "El fitxer comprimit (format .tgz o .tar.gz) NO EXISTEIX !!!"
124         echo "$(tput setaf 1)*****      ERROR      *****$(tput sgr0)"
125         echo ""
126         echo "$(tput setaf 2)          /\ /\          $(tput sgr0)"
127         echo "$(tput setaf 3)          |||| |          $(tput sgr0)"
128         echo "$(tput setaf 4)          \ | \          $(tput sgr0)"
129         echo "$(tput setaf 5)          _ _ / ()() ERROR      $(tput sgr0)"
130         echo "$(tput setaf 6)          /   \  =>*<=          $(tput sgr0)"
131         echo "$(tput setaf 7)          /|      \  /|          $(tput sgr0)"
132         echo "$(tput setaf 8)          \|      /_ | |          $(tput sgr0)"
133         echo "$(tput setaf 9)          \____) \_)          $(tput sgr0)"
134         echo "$(tput setaf 1)*****$(tput sgr0)"
135         mostrar_menu
136         exit 1;
137     fi
138
139 fi

```

3. JOCS DE PROVES.

3.1. JOCS DE PROVES Copia.sh

Execució sense cap parametre:

Deuria mostrar un missatge d'ús.

```
alfonso@alfonso-1-2:~/Documentos/SCRIPT COPIA$ ./copia.sh
Ús: copia.sh [-h] [-K] [-D] [arxius/directoris]

On:
-h Mostra aquest missatge
-K Comprova si fitxer esta al comprimit, si esta, afegeix el nou amb la data d'ahui.
-D DryRun, no farà res, sols mostra per pantalla les execucions que faria el script.

Exemples:

> Afegir arxius mes nous que les de la copia
copia.sh -K arxiu_3 arxiu_2 dir_1

> Mostrar tot el que farà el script sense modificar res
copia.sh -D dir_1 arxiu_4

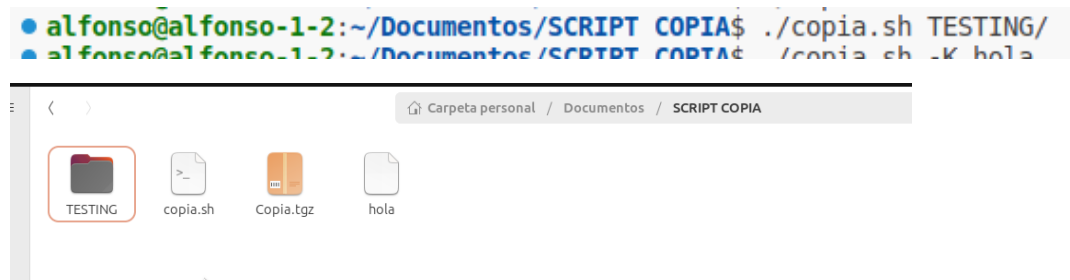
> Generar una copia de seguretat si no existeix cap o afegir a la copia actual si existeix
copia.sh arxiu_3 arxiu_4 arxiu_5 dir_3
```

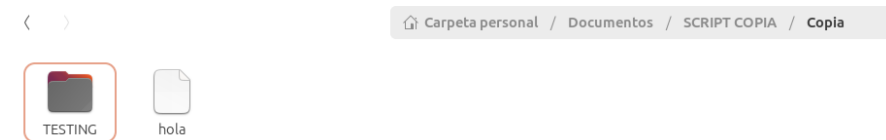
Execució amb un arxiu de parametre:

Genera un arxiu .tgz amb el arxiu dintre comprimit.



Execució per afegir una carpeta al arxiu creat en el pas anterior:

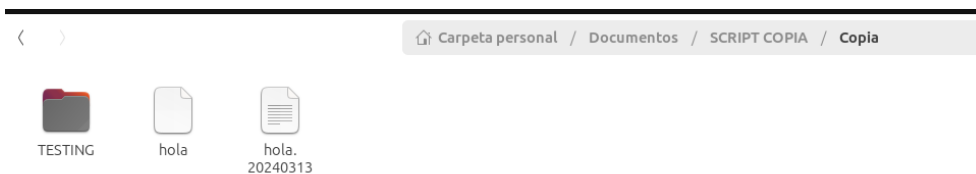




Si modifiquem el arxiu hola, vorem que tindrem 2, el original i un nou amb la data d'ahui.

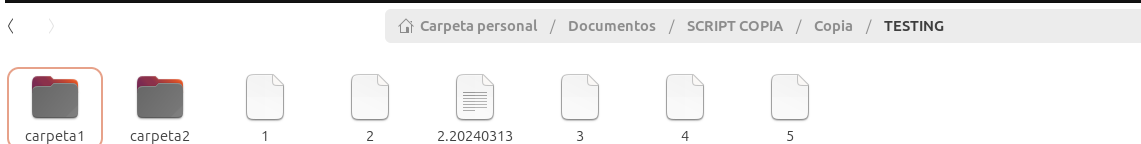
```
alfonso@alfonso-1-2:~/Documentos/SCRIPT COPIA$ ./copia.sh -K hola
```

Una vegada descomprimit el tgz.



Si modifiquem el arxiu 2 dintre de Testing, vorem que funciona també perfectament.

```
alfonso@alfonso-1-2:~/Documentos/SCRIPT COPIA$ ./copia.sh -K TESTING/2
```



Si fem un dryrun vorem el que faria el script, sense modificar res:

```
alfonso@alfonso-1-2:~/Documentos/SCRIPT COPIA$ ./copia.sh -D hola TESTING
Afeint > (f)hola a llista fitxers
Afeint > (d)TESTING a llista directoris.
Se ha afeint la llista generada al archiu Copia.tgz!!
alfonso@alfonso-1-2:~/Documentos/SCRIPT COPIA$
```

Si posem un argument erroni, vorem que tenim un missatge d'error.

[illegible]

Finalment, per tractar correctament les rutes absolutes, he fet que sempre el tgz es cree en una carpeta Copia de la carpeta de usuari. Aquesta carpeta es pot definir fàcilment modificant la variable del script.

```
fullpath="/home/milax/COPIA"
```

Quedant així el següent:




3.2. JOCS DE PROVES Recupera.sh

Execució sense cap paràmetre:

```
evericat@edu-PC: /mnt/c/Users/veric/Desktop/RECUPERAR$ ./recuperar.sh  
ERROR: Par metres insuficients !!!  
*****  
  
      ^ ^  
    ||| ||  
   \  |  \  
  -  |  -  
 / \  |  / \ C C ERROR  
/_ \|  |  _/\ =>*<=  
 \|  |  \|  |  
  \--|  --|  
   \---)  \---)  
*****
```

Execució sense que existeixi l'arxiu comprimit:

```
evericat@edu-PC: /mnt/c/Users/veric/Desktop/RECUPERAR$ ./recuperar.sh Copias.tgz a b c desti
El fitxer comprimit (format .tgz o .tar.gz) NO EXISTEIX !!!
***** ERROR *****
```



```
*****
./recuperar.sh: line 143: mostrar_menu: command not found
evericat@edu-PC: /mnt/c/Users/veric/Desktop/RECUPERAR$ |
```

Execució en mode DryRun:


The image shows a Windows desktop environment. On the left is a terminal window titled 'evericat@edu-PC: /mnt/c/Users...' with the following command history:

```
evericat@edu-PC:/mnt/c/Users/veric/Desktop/RECUPERAR$ ./recuperar.sh -D Copia.tgz pepe.txt TESTING desti
Executant en mode de simulació (Dry Run)...
Creant carpeta DESTI amb el nom desti...
Copiant directori /tmp/tmp.CT7xGfNFHQ/TESTING a desti...
Copiant /tmp/tmp.CT7xGfNFHQ/pepe.txt a desti...
Copiant /tmp/tmp.CT7xGfNFHQ/pepe.txt.28248213 a desti...
evericat@edu-PC:/mnt/c/Users/veric/Desktop/RECUPERAR$
```

On the right is a File Explorer window titled 'RECUPERAR' showing the 'Desktop' location. It displays two files: 'Copia.tgz' and 'recuperar.sh', both with a date modified of 15/03/2024 15:23.

Podem observar com ha mostrat tots els passos però no ha fet res ni ha creat cap directori.

Execució normal per a recuperar el directori TESTING i el fitxer pepe.txt.

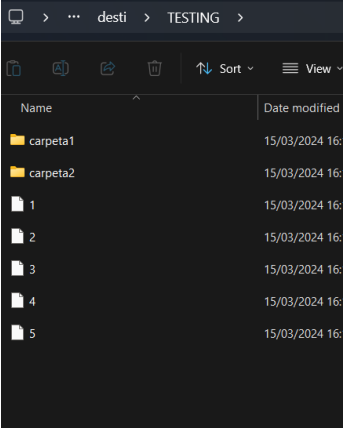
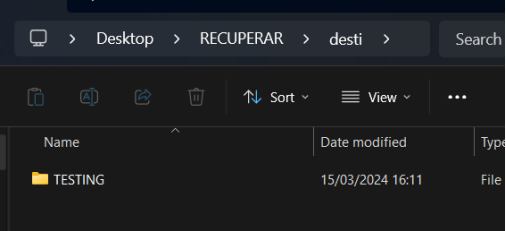


The screenshot shows a terminal window on the left and a file explorer on the right. In the terminal, the user runs the command `./recuperar.sh Copia.tgz pepe.txt TESTING dest`. The file explorer on the right shows the contents of the `dest` directory, which now includes a `TESTING` folder, a `pepe.txt` file, and a `pepetxt20240911` file, all dated 15/09/2024 at 15:37.

Podem veure que també ha recuperat el fitxer pepe.txt.20240313

A screenshot of a Windows File Explorer window. The title bar says 'Recuperar'. The address bar shows the path 'desti'. The search bar contains 'Search desti'. The ribbon has icons for 'File', 'Share', 'Move', 'Delete', 'Sort', 'View', and 'Details'. The file list shows three items: a folder named 'TESTING', a file named 'pepe.txt', and a file named 'pepe.txt.20240313'. The columns are 'Name', 'Date modified', and 'Type'.

Execuci



4. EXECUCIO CRONTAB TASCA COPIA.SH

Ens assegurem primer que l'arxiu copia.sh tingui permisos d'execució.

```
copia.sh |
```

Si no el té fem servir el comando `sudo chmod +x archivo.sh`

A més modifiquem el path de la carpeta on es guardés la còpia:

```
fullpath="/home/milax/COPIA"
```

Ara procedim a editar el crontab.

```
milax@casa:~/SCRIPTS$ crontab -e
```

Afegim la següent línia:

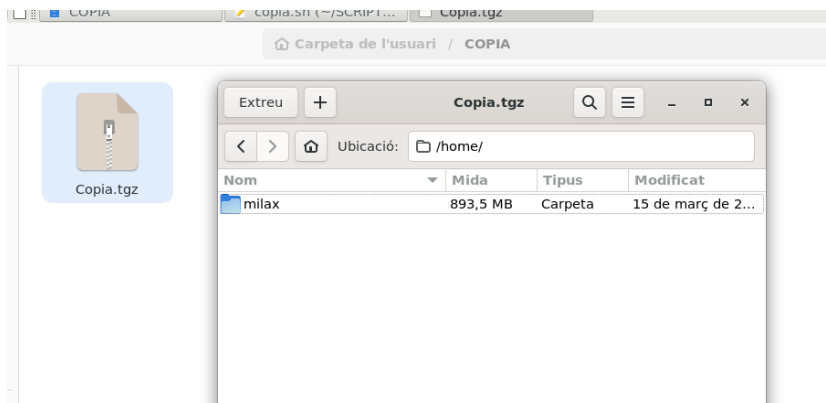
```
# m h dom mon dow command
*/1 * * * * /home/milax/SCRIPTS/copia.sh /home/milax/
```

Aquesta línia executarà l'script cada 1m. Això es fa primerament per poder provar que tot funciona correctament. Més endavant assignarem un temps adequat.

Reiniciem el servei de crontab:

```
milax@casa:~/SCRIPTS$ sudo service cron restart
```

Després d'això, esperem 1m a que s'executi l'script, obtenint el següent resultat:



Ara modifiquem el crontab perquè s'executi per exemple, cada dia:

```
# m h dom mon dow command
3 0 * * * /home/milax/SCRIPTS/copia.sh /home/milax/
```

Això executarà l'script tots els dies a les 3 del matí.

També com a anotació considerem que per a aquesta tasca és més convenient fer ús de anacron. Ja que crontab no s'executarà si el ordinador està apagat a l'hora de la execució de la tasca i després l'engegem al matí. Anacron sí que guarda aquesta informació i si durant el dia alguna tasca no s'ha executat, la executa immediatament quan engegem el ordinador. Asegurant així una còpia diària de les dades.

