FONAMENTS DE SISTEMES OPERATIUS

Pràctica 1: Comparador de directoris avançat

UNIVERSITAT ROVIRA i VIRGILI

Curs 2024-2025

Grau d'Enginyeria Informàtica

Integrants

Álvaro Pérez Caballer

Eloi Viciana Gómez

Professorat:

Carles Aliagas Castell

Maria dels Àngels Moncusí Mercadé

Taula de continguts

Estructura del projecte

Aquest projecte implementa un script de Bash per comparar directoris de manera avançada, amb diverses funcionalitats com comparació recursiva, detecció de similitud entre fitxers, comprovació de permisos i més.

Per tal d'assegurar el seu correcte funcionament, cal primer formalitzar l'estructura que seguiran els fitxers, assegurant així la màxima eficiència possible per als futurs joc de proves que es realitzaran.

Per tal motiu, a més a més de facilitar la col·laboració entre els diferents components del grup de la pràctica, s'ha dissenyat la següent estructura de directoris:

```
P1-FSO/
├── README.md                    # Aquest fitxer
├── scripts/
│     ├── comparador_directoris.sh #Script principal amb les funcionalitats
implementades
│   ├── crear_proves.sh          # Script per crear l'entorn de proves
│   └── scriptInicial.sh         # Versió inicial del script (opcional)
├── joc_proves/                  # Directori amb els jocs de proves
│   ├── dir1/                    # Primer directori de proves
│   │   ├── backup.bak
│   │   ├── diferent1.txt
│   │   ├── executable.txt
│   │   ├── fitxer_gran.txt
│   │   ├── ignorar/
│   │   │   └── ignorar_file.txt
│   │   ├── ignorar2/
│   │   │   └── ignorar2_file.txt
│   │   ├── igual1.txt
│   │   ├── igual2.txt
│   │   ├── mateix_nom.txt
│   │   ├── permisos.txt
│   │   ├── similar1.txt
│   │   ├── similar_90.txt
│   │   ├── subdir/
│   │   │   ├── sub_diff.txt
│   │   │   ├── sub_igual.txt
│   │   │   └── unic_sub1.txt
│   │   └── temporal.tmp
│   ├── dir2/                    # Segon directori de proves
│   │   ├── backup.bak
│   │   ├── diferent2.txt
│   │   ├── executable.txt
│   │   ├── fitxer_gran.txt
│   │   ├── ignorar/
│   │   │   └── ignorar_file.txt
```

```
│   │   ├── ignorar2/
│   │   │   └── ignorar2_file.txt
│   │   ├── igual1.txt
│   │   ├── igual2.txt
│   │   ├── mateix_nom.txt
│   │   ├── permisos.txt
│   │   ├── similar2.txt
│   │   ├── similar_90.txt
│   │   ├── subdir/
│   │   │   ├── sub_diff.txt
│   │   │   ├── sub_igual.txt
│   │   │   └── unic_sub2.txt
│   │   └── temporal.tmp
│   └
├── resultats.txt               # Exemple de fitxer de sortida
└── doc/
    └── documentacio.pdf # Documentació completa de la pràctica
```

Es destaquen els continguts següents:

- **Fitxer README:** fitxer resum en estil de Markdown, explicatiu tan del contingut del projecte i el seu contingut bàsic com de les instruccions per poder executar el script.

- **Directori "scripts":** conté l'script final de la pràctica juntament amb un script dissenyat en inicialitzar el nostre joc de proves.

- **Directori "joc_proves":** conté els dos directoris emprats per als jocs de proves, amb tots els fitxers i subdirectoris necessaris per assegurar la seva correctesa.

- **Directori "doc":** amb el contingut d'aquesta mateixa documentació i l'enunciat de la pràctica.

Script complet “comparador_directoris.sh”

```
#!/bin/bash
# Script per comparar dos directoris, mostrant els fitxers únics
# i els que tenen el mateix nom però són diferents.
# Inclou funcionalitats avançades com la similitud entre fitxers
# i ignorar determinats fitxers i permisos.

################################################################################
#
### CONFIGURACIÓ I VARIABLES GLOBALS ###########################################
################################################################################
#
DIFF_OPTS="-w -B -u"
# DIFF_OPTS defineix les opcions que fem servir per a la comanda 'diff':
#   -w : ignora canvis en espais en blanc
#   -B : ignora línies completament en blanc
#   -u : mostra el format "unificat" en les diferències

SIMILARITY_THRESHOLD="90"
# Percentatge de similitud per defecte, emprat per la funció check_similarity

IGNORE_EXTS=""
# Llista d'extensions a ignorar, separades per comes (ex: ".tmp,.bak"), que
s'usaran a 'build_find_command'

IGNORE_SUBDIR=""
# Nom d'un subdirectori a ignorar (ex: "ignorar"), també s'exclou amb
'build_find_command'

CHECK_PERMS=false
# Indica si cal comparar permisos dels fitxers (paràmetre -p)

OUTPUT_FILE=""
# Si val "" (buit), la sortida va a pantalla; si té un nom de fitxer, la
sortida s'hi redirigeix (paràmetre -o)

RECURSIVE=false
# Indica si la comparació s'ha de fer recursivament (paràmetre -r, per entrar
dins subdirectoris)

################################################################################
#
### FUNCIÓ D'AJUDA PER MOSTRAR ÚS DE L'SCRIPT
#################################
```

```
################################################################################
#
usage() {
    # Mostra un text d'ajuda explicant les opcions i la seva sintaxi
    echo "Ús: $0 [-r] [-s <similitud>] [-e <ext1,ext2,...>] [-d <subdir>] [-p]
[-o <output>] <directori1> <directori2>" >&2
    #  -r: indica cerca recursiva
    #  -s: percentatge de similitud que volem per considerar dos fitxers
"similars"
    #  -e: llista d'extensions separades per comes a ignorar
    #  -d: subdirectori a ignorar
    #  -p: compara també els permisos
    #  -o: fitxer de sortida on es registrarà tota la informació
    echo "  -r: Cerca recursiva en subdirectoris" >&2
    echo "  -s <valor>: Percentatge de similitud (defecte: 90)" >&2
    echo "  -e <ext1,ext2,...>: Extensions a ignorar" >&2
    echo "  -d <subdir>: Subdirectori a ignorar" >&2
    echo "  -p: Comparar permisos" >&2
    echo "  -o <fitxer>: Fitxer de sortida" >&2
    exit 1
    # Finalitza mostrant que s'ha produït un "error" o que manca info
}

################################################################################
#
### PROCESSAMENT D'ARGUMENTS
#####################################################
################################################################################
#
# Amb getopts capturem les opcions -r, -s, -e, -d, -p i -o
# i les assignem a les variables corresponents.
while getopts ":rs:e:d:po:" opt; do
    case "$opt" in
        r) RECURSIVE=true ;;                        # -r => cerca recursiva
        s) SIMILARITY_THRESHOLD="$OPTARG" ;;         # -s => llindar de
similitud
        e) IGNORE_EXTS="$OPTARG" ;;                  # -e => extensions a
ignorar
        d) IGNORE_SUBDIR="$OPTARG" ;;                # -d => subdirectori a
ignorar
        p) CHECK_PERMS=true ;;                       # -p => compara permisos
        o) OUTPUT_FILE="$OPTARG" ;;                  # -o => fitxer de sortida
        \?) echo "Opció no reconeguda: -$OPTARG" >&2; usage ;;  # Error
d'opció
        :) echo "L'opció -$OPTARG requereix un valor." >&2; usage ;; # Falta
valor
    esac
```

```
done

shift $((OPTIND - 1))
# Elimina els paràmetres ja processats per getopts, de manera que
# només quedin els paràmetres obligatoris (en aquest cas, <dir1> i <dir2>)

################################################################################
#
### VALIDACIÓ D'ARGUMENTS
#####################################################
################################################################################
#
if [ "$#" -ne 2 ]; then
    # Si, després de processar, no hi ha exactament 2 paràmetres,
    # mostrem l'ajuda i sortim.
    usage
fi

DIR1=$1
DIR2=$2
# Agafem els dos directors restants

# Comprova si els directoris existeixen
if [ ! -d "$DIR1" ] || [ ! -d "$DIR2" ]; then
    # Si un o ambdós no són directoris, mostrem error i sortim
    echo "Un o ambdós directoris no existeixen." >&2  # Aquesta sortida va a
pantalla (error)
    echo "" >&2
    exit 1
fi

################################################################################
#
### FUNCIONS D'UTILITAT
#########################################################
################################################################################
#
echo -n "" > "$OUTPUT_FILE"

output() {
    # Funció que, en lloc d'usar 'echo' directe, envia la sortida
    # a un fitxer (si OUTPUT_FILE no és buit) o per pantalla
    # (si no s'ha especificat -o).
    if [ -n "$OUTPUT_FILE" ]; then
        echo "$1" >> "$OUTPUT_FILE"
    else
```

```
        echo "$1"
    fi
}

build_find_command() {
    # Construeix la comanda 'find' per llistar fitxers
    # segons si és recursiva, si cal ignorar subdirectori,
    # o extensions concretes.
    local dir="$1"
    local cmd="find \"$dir\""

    if [ "$RECURSIVE" = false ]; then
        # Si NO és recursiu, limitem la cerca a -maxdepth 1
        cmd="$cmd -maxdepth 1"
    fi

    cmd="$cmd -type f"
    # Només fitxers regulars

    if [ -n "$IGNORE_SUBDIR" ]; then
        # Si tenim subdirectori a ignorar,
        # indiquem a find que no agafi cap fitxer
        # dins la ruta "*/$IGNORE_SUBDIR/*"
        cmd="$cmd -not -path \"*/$IGNORE_SUBDIR/*\""
    fi

    if [ -n "$IGNORE_EXTS" ]; then
        # Si hi ha extensions a ignorar, convertim la llista separada per
comes
        # en múltiples condicions '-not -name "*.<ext>"'
        IFS=',' read -r -a exts_array <<< "$IGNORE_EXTS"
        for ext in "${exts_array[@]}"; do
            local trimmed_ext="$(echo "$ext" | xargs)"
            # xargs elimina espais sobrants al voltant
            cmd="$cmd -not -name \"*.$trimmed_ext\""
        done
    fi

    echo "$cmd"
    # Retornem la cadena amb la comanda 'find' completa
}

check_similarity() {
    # Compara la semblança entre dos fitxers, calculant un % de similitud
    # i si aquest % >= SIMILARITY_THRESHOLD, mostra un missatge.
    local f1="$1"
```

```
    local f2="$2"

    # Comptem el nombre de línies no buides de cada fitxer
    local lines1=$(grep -vE '^[[:space:]]*$' "$f1" | wc -l)
    local lines2=$(grep -vE '^[[:space:]]*$' "$f2" | wc -l)
    local max_lines=$(( lines1 > lines2 ? lines1 : lines2 ))
    # max_lines és el màxim entre lines1 i lines2

    # Fem servir 'diff' per veure quantes línies realment difereixen.
    # grep '^[+-]' filtra les línies que comencen amb + o - al format
unificat.
    local diff_lines=$(diff -u $DIFF_OPTS "$f1" "$f2" 2>/dev/null | grep '^[+-
]' | wc -l)

    # Calculem la similitud com:
    #   ((max_lines - diff_lines) / max_lines) * 100
    local similarity=$(echo "scale=2; if($max_lines>0) (($max_lines -
$diff_lines)/$max_lines)*100 else 0" | bc)

    # Compareu la 'similarity' amb el llindar definit (SIMILARITY_THRESHOLD)
    local compare=$(echo "$similarity >= $SIMILARITY_THRESHOLD" | bc -l)
    if [ "$compare" -eq 1 ]; then
        # Si és >=, enviem el missatge
        output "El fitxer \"$(readlink -f "$f1")\" té una similitud del
${similarity}% amb \"$(readlink -f "$f2")\""
    fi
}

comparar_permissos() {
    # Comprova si dos fitxers tenen els mateixos permisos
    # i, si no és així, ho mostra.
    local fitxer1="$1"
    local fitxer2="$2"

    # Obtenim els permisos en format octal (ex: 644, 755...)
    local permis1=$(stat -c "%a" "$fitxer1")
    local permis2=$(stat -c "%a" "$fitxer2")

    if [ "$permis1" != "$permis2" ]; then
        # Si són diferents, ho escrivim a la sortida (fitxer o pantalla)
        output "Permisos diferents per $fitxer1 i $fitxer2:"
        output "$fitxer1: $permis1"
        output "$fitxer2: $permis2"
    fi
}
```

```
################################################################################
#
### COMPARACIÓ DE FITXERS ÚNICS
################################################
################################################################################
#
# Ara, amb 'comm', trobem els fitxers que només són a DIR1 i a DIR2
# (comparant noms relatius).
output ""
output "Fitxers només a $DIR1:"

cmdFindDir1=$(build_find_command "$DIR1")
cmdFindDir2=$(build_find_command "$DIR2")
# Guardem en dues variables les comandes per trobar fitxers a cada directori

comm -23 <( eval "$cmdFindDir1" | sed "s|^$DIR1/||" | sort ) \
         <( eval "$cmdFindDir2" | sed "s|^$DIR2/||" | sort ) | while read -r
file; do
    # 'comm -23' treu només les línies presents a la primera llista i no a la
segona
    # - Els <(...) són substitucions de procés; executem 'eval "$cmdFindDir1"'
    #   i ens quedem la seva sortida, transformant la ruta absoluta a una ruta
    #   relativa amb sed "s|^$DIR1/||"
    # - A sort final, comparem les dues llistes i la diferència surt per la
canal
    output "  - $file"
done

output ""
output "Fitxers només a $DIR2:"

comm -13 <( eval "$cmdFindDir1" | sed "s|^$DIR1/||" | sort ) \
         <( eval "$cmdFindDir2" | sed "s|^$DIR2/||" | sort ) | while read -r
file; do
    # 'comm -13' treu només les línies de la segona llista (no a la primera).
    output "  - $file"
done

################################################################################
#
### COMPARACIÓ DETALLADA DE FITXERS
##########################################
################################################################################
#
# En aquest bloc, mirem quins fitxers (a DIR1) també existeixen a DIR2,
# i, si són diferents, mostrem la comparació de 'diff'.
eval "$cmdFindDir1" | while read -r file; do
```

```
    # Per a cadascun dels fitxers trobats a DIR1
    relative_path="${file#$DIR1/}"
    # El camí relatiu (traient la part /path/dir1/)

    if [ -f "$DIR2/$relative_path" ]; then
        # Si el fitxer també existeix a DIR2 amb el mateix nom relatiu
        if ! diff -q $DIFF_OPTS "$file" "$DIR2/$relative_path" > /dev/null
2>&1; then
            # diff -q : mode ràpid => si ret 1, hi ha diferències
            # '!' => invertim el resultat: entrem aquí si 'diff -q' troba
diferències
            output ""
            output "Fitxer diferent: $relative_path"
            output "----- Diferències ($file) vs ($DIR2/$relative_path) -----"
            # Mostrem totes les diferències amb el format complet
            diff $DIFF_OPTS "$file" "$DIR2/$relative_path" | while read -r
line; do
                output "$line"
            done
        fi

        # Si s'ha demanat comparar permisos (CHECK_PERMS=true)
        if [ "$CHECK_PERMS" = true ]; then
            comparar_permissos "$file" "$DIR2/$relative_path"
            output ""
        fi
    fi
done

#############################################################################
#
### COMPARACIÓ DE SIMILITUD ENTRE TOTS ELS FITXERS
#############################
#############################################################################
#
# Compara la similitud (>= SIMILARITY_THRESHOLD) entre TOTS els fitxers
# de DIR1 i DIR2 (no sols aquells que tinguin el mateix nom).
output ""
output "Comparació de similitud entre tots els fitxers (>=
$SIMILARITY_THRESHOLD%):"
eval "$cmdFindDir1" | while read -r file1; do
    eval "$cmdFindDir2" | while read -r file2; do
        # Si no són el mateix fitxer (ruta absoluta) ni el mateix nom relatiu,
        # aleshores intentem comprovar si hi pot haver similitud de contingut.
        # Això permet descobrir, per exemple, si 'similar1.txt' i
'similar2.txt'
        # tenen un 95% de semblança (encara que tinguin noms diferents).
```

```
        if [ "$file1" != "$file2" ] || [ "${file1#$DIR1/}" !=
"${file2#$DIR2/}" ]; then
            check_similarity "$file1" "$file2"
        fi
    done
done
```

Funcionalitat 1: Cerca recursiva en subdirectoris

Objectiu

L'script pot comparar únicament els fitxers directes del directori (nivell 1) o, si s'activa el mode **recursiu**, tots els fitxers de tots els subdirectoris que hi hagi dins de les carpetes que es comparen.

Com està implementada

- **Variable global**: `RECURSIVE=false`, indica que per defecte no es fa la cerca recursiva.
- **Paràmetre -r**: quan l'usuari l'especifica a la línia d'ordres, es canvia a `RECURSIVE=true`.
- **Funció `build_find_command()`**:
  - Si `RECURSIVE` és `false`, afegeix l'opció `-maxdepth 1` a la comanda `find`, de manera que no entrarà en subcarpetes.
  - Si `RECURSIVE` és `true`, **no** posa el `-maxdepth`, així que `find` visitarà totes les profunditats de subdirectoris.

Ara, per a cada subdirectori dins `dirA` i `dirB`, `find` continuarà buscant fitxers; això permet detectar coincidències o diferències que estiguin amagades en carpetes internes.

Aquesta opció és essencial quan es vol una comparació **completa** i no només superficial. Permet, per exemple, detectar fitxers asíncrons o únics en rutes internes.

Funcionalitat 2: Similitud entre fitxers

Objectiu

Analitzar **quant** (en percentatge) s'assemblen dos fitxers, encara que no tinguin el mateix nom, i mostrar un missatge si aquesta semblança supera un **llindar** determinat (per defecte, un **90%**).

Com està implementada

- **Paràmetre -s <valor>**: permet indicar quin percentatge de similitud es consideraria suficient per informar-se. Si no es dona, s'utilitza la variable per defecte `SIMILARITY_THRESHOLD="90"`.

- **Funció `check_similarity(f1, f2)`**:
  - **Compte** quantes línies **no buides** té cada fitxer, usant `grep -vE '^[[:space:]]*$' i wc -l`.
  - **Determina** la quantitat de línies diferents amb `diff -u`, i filtrant aquelles que comencen en + o - (que indiquen canvis). Aquesta és la variable `diff_lines`.
  - **Càlcul**:

    $$\text{similarity } (\%) = \left(\frac{\text{max\_lines} - \text{diff\_lines}}{\text{max\_lines}} \right) \times 100$$

    on `max_lines` és el nombre més gran de línies no buides entre els dos fitxers.
  - **Comparació**: amb `bc -l` mirem si la `similarity` és major o igual que el llindar `SIMILARITY_THRESHOLD`. Si ho és, s'imprimeix un missatge, indicant la similitud i les rutes reals dels fitxers.

- **Bucle de comparació**:
  Després de fer la comparació normal de fitxers amb el mateix nom, l'script re-corre **tots** els fitxers del directori A i **tots** del directori B, cridant `check_similarity` per a cada combinació. Això fa que, fins i tot si dos fitxers tenen noms diferents (p. ex. `similar1.txt` i `similar2.txt`), es pugui detectar que tenen un 95% de semblança.

Funcionalitat 3: Ignorar determinats fitxers

Objectiu

Excloure de la comparació:

- Fitxers amb **extensions** concretes (p. ex. `.tmp`, `.bak`).
- **Un subdirectori** complet (p. ex. `ignorar`), de manera que l'script no consideri res que hi hagi dins.

Com està implementada

- **Paràmetres**:
  - `-e <ext1,ext2,...>`: rep una llista separada per comes, com `".tmp,.bak"`.
  - `-d <subdir>`: indica un nom de carpeta, com `ignorar`.

- **Funció `build_find_command()`**:
  - Inicia amb `find "$dir"` (i potser `-maxdepth 1` si no és recursiu).
  - Filtra només fitxers (`-type f`).
  - **Subdirectori**: si `IGNORE_SUBDIR` no és buit, afegeix `-not -path "*/$IGNORE_SUBDIR/*"`. Això evita qualsevol fitxer que es trobi dins la carpeta "$IGNORE_SUBDIR/…".
  - **Extensions**: es llegeix `IGNORE_EXTS`, se separa per comes, i per a cadascuna s'afegeix `-not -name "*.<ext>"`. Així, per exemple, `-not -name "*.bak"`.

Quan s'executen aquestes comandes `find`, tots els fitxers que compleixin els criteris d'ignorància **són exclosos** i no apareixeran en cap comparació posterior (ni per `comm`, ni per `diff`, etc.)

Utilitat

És molt útil per no embrutar els resultats amb fitxers temporals, de còpia de seguretat, o directoris on no ens interessi. Així fem una comparació més neta.

Funcionalitat 4: Comprovació de permisos

Objectiu

Quan dos fitxers coincideixen en **nom** (és a dir, rutes relatius iguals a DIR1 i DIR2), a més d'analitzar-ne el contingut, també es pot decidir comprovar si tenen els **mateixos permisos** de fitxer (com per exemple 755, 644, etc.).

Com està implementada

- **Bandera**: `CHECK_PERMS=false`, activada amb l'opció `-p`.
- **Funció `comparar_permissos(fitxer1, fitxer2)`**:
  - Obté els permisos en octal (`stat -c "%a"`) de cadascun.
  - Els compara. Si **no** coincideixen, mostra un missatge detallant la discrepància.
- **Ubicació** en el codi:
  - Dins el bucle on es comprova si el fitxer **existeix** en ambdós directoris, just després de verificar si és igual o no en contingut.
  - Si `CHECK_PERMS` és `true`, es crida la funció. En cas contrari, s'omet.

Funcionalitat 5: Registre en un fitxer

Objectiu

En comptes d'enviar la sortida directament per pantalla, donar l'opció de **guardar** tots els resultats (fitxers únics, diferències, similituds, permisos, etc.) en un **fitxer** que l'usuari triï.

Com està implementada

- **Paràmetre**: `-o <output_file>`. Desa el camí del fitxer en la variable `OUTPUT_FILE`.
- **Funció `output()`**:
  - Rep un **missatge** a escriure.
  - Si `OUTPUT_FILE` està en blanc (`""`), fa un simple `echo` (per pantalla).

- Si té un valor (p. ex. `resultats.txt`), fa `echo "..." >> resultats.txt`, de manera que escriu al **fitxer** en mode append.

La comparació s'executa, i tot allò que l'script mostraria normalment per pantalla queda guardat en `resultats.txt`.

Instruccions d'execució

1. Preparació de l'entorn

El projecte inclou un **script** anomenat (per exemple) `crear_proves.sh` que s'encarrega de generar tot un **joc de proves** dins dels directoris `joc_proves/dir1` i `joc_proves/dir2`. Això permet fer tests **realistes** sobre la comparació de fitxers sense haver de preparar manualment cada situació possible (fitxers iguals, permisos diferents, extensions a ignorar, etc.).

SCRIPT:

```
# Script per crear un joc de proves.
# Netegem els directoris anteriors, si ja existeixen.
# Es fa recursivament per eliminar subdirectoris.
# Eliminem la variable per evitar conflictes
unset BASE_DIR
# Creem el directori base per al joc de proves
# Veiem la ruta relativa de l'script, és a dir, on s'executa.
SCRIPT_DIR="$(dirname "$0")"
# Agafem el path absolut, i anem nivell enrere
BASE_DIR="$(realpath "$SCRIPT_DIR/..")"

rm -rf "$BASE_DIR/joc_proves"

# Creem l'estructura de directoris dins de joc_proves
mkdir -p "$BASE_DIR/joc_proves/dir1/subdir"
mkdir -p "$BASE_DIR/joc_proves/dir2/subdir"
mkdir -p "$BASE_DIR/joc_proves/dir1/ignorar"
mkdir -p "$BASE_DIR/joc_proves/dir2/ignorar"
mkdir -p "$BASE_DIR/joc_proves/dir1/ignorar2"
mkdir -p "$BASE_DIR/joc_proves/dir2/ignorar2"

# Fitxers iguals en ambdós directoris
echo "Aquest contingut és exactament igual" >
"$BASE_DIR/joc_proves/dir1/igual1.txt"
echo "Aquest contingut és exactament igual" >
"$BASE_DIR/joc_proves/dir2/igual1.txt"
echo "Un altre contingut igual" > "$BASE_DIR/joc_proves/dir1/igual2.txt"
echo "Un altre contingut igual" > "$BASE_DIR/joc_proves/dir2/igual2.txt"

# Fitxers amb el mateix nom però diferent contingut
echo "Contingut al directori 1" > "$BASE_DIR/joc_proves/dir1/mateix_nom.txt"
echo "Contingut al directori 2" > "$BASE_DIR/joc_proves/dir2/mateix_nom.txt"

# Fitxers únics a cada directori
```

```
echo "Aquest fitxer només existeix a dir1" >
"$BASE_DIR/joc_proves/dir1/diferent1.txt"
echo "Aquest fitxer només existeix a dir2" >
"$BASE_DIR/joc_proves/dir2/diferent2.txt"

# Fitxers amb contingut similar però no idèntic (80% similitud)
echo "Línia 1 - Igual" > "$BASE_DIR/joc_proves/dir1/similar1.txt"
echo "Línia 2 - Igual" >> "$BASE_DIR/joc_proves/dir1/similar1.txt"
echo "Línia 3 - Igual" >> "$BASE_DIR/joc_proves/dir1/similar1.txt"
echo "Línia 4 - Només a l'arxiu 1" >> "$BASE_DIR/joc_proves/dir1/similar1.txt"
echo "Línia 5 - Igual" >> "$BASE_DIR/joc_proves/dir1/similar1.txt"

echo "Línia 1 - Igual" > "$BASE_DIR/joc_proves/dir2/similar2.txt"
echo "Línia 2 - Igual" >> "$BASE_DIR/joc_proves/dir2/similar2.txt"
echo "Línia 3 - Igual" >> "$BASE_DIR/joc_proves/dir2/similar2.txt"
echo "Línia 4 - Només a l'arxiu 2" >> "$BASE_DIR/joc_proves/dir2/similar2.txt"
echo "Línia 5 - Igual" >> "$BASE_DIR/joc_proves/dir2/similar2.txt"

# Fitxers amb similitud exactament del 90%
echo -e "Línia 1\nLínia 2\nLínia 3\nLínia 4\nLínia 5\nLínia 6\nLínia 7\nLínia
8\nLínia 9\nLínia 10 - Diferent" > "$BASE_DIR/joc_proves/dir1/similar_90.txt"
echo -e "Línia 1\nLínia 2\nLínia 3\nLínia 4\nLínia 5\nLínia 6\nLínia 7\nLínia
8\nLínia 9\nLínia 10 - Diferent a dir2" >
"$BASE_DIR/joc_proves/dir2/similar_90.txt"

# Fitxers amb extensions per provar el filtrat
echo "Contingut temporal" > "$BASE_DIR/joc_proves/dir1/temporal.tmp"
echo "Contingut de backup" > "$BASE_DIR/joc_proves/dir1/backup.bak"
echo "Contingut temporal" > "$BASE_DIR/joc_proves/dir2/temporal.tmp"
echo "Contingut de backup" > "$BASE_DIR/joc_proves/dir2/backup.bak"

# Fitxers en subdirectoris (per provar recursivitat)
echo "Contingut en subdirectori igual" >
"$BASE_DIR/joc_proves/dir1/subdir/sub_igual.txt"
echo "Contingut en subdirectori igual" >
"$BASE_DIR/joc_proves/dir2/subdir/sub_igual.txt"
echo "Contingut diferent subdir1" >
"$BASE_DIR/joc_proves/dir1/subdir/sub_diff.txt"
echo "Contingut diferent subdir2" >
"$BASE_DIR/joc_proves/dir2/subdir/sub_diff.txt"

# Fitxers únics en subdirectoris
echo "Únic en subdir1" > "$BASE_DIR/joc_proves/dir1/subdir/unic_sub1.txt"
echo "Únic en subdir2" > "$BASE_DIR/joc_proves/dir2/subdir/unic_sub2.txt"

# Fitxers en directori a ignorar
```

```
echo "Aquest directori hauria d'ignorar-se" >
"$BASE_DIR/joc_proves/dir1/ignorar/ignorar_file.txt"
echo "Aquest directori hauria d'ignorar-se" >
"$BASE_DIR/joc_proves/dir2/ignorar/ignorar_file.txt"
echo "Fitxer a ignorar2" >
"$BASE_DIR/joc_proves/dir1/ignorar2/ignorar2_file.txt"  # Afegit per simetria
echo "Fitxer a ignorar2" >
"$BASE_DIR/joc_proves/dir2/ignorar2/ignorar2_file.txt"

# Fitxers amb diferents permisos
echo "Fitxer amb permisos diferents" >
"$BASE_DIR/joc_proves/dir1/permisos.txt"
echo "Fitxer amb permisos diferents" >
"$BASE_DIR/joc_proves/dir2/permisos.txt"
chmod 644 "$BASE_DIR/joc_proves/dir1/permisos.txt"
chmod 755 "$BASE_DIR/joc_proves/dir2/permisos.txt"

# Segon cas de permisos diferents
echo "Executable vs no executable" >
"$BASE_DIR/joc_proves/dir1/executable.txt"
echo "Executable vs no executable" >
"$BASE_DIR/joc_proves/dir2/executable.txt"
chmod 600 "$BASE_DIR/joc_proves/dir1/executable.txt"
chmod 700 "$BASE_DIR/joc_proves/dir2/executable.txt"

# Fitxer gran per provar redirecció
for i in {1..100}; do echo "Línia $i d'un fitxer gran" >>
"$BASE_DIR/joc_proves/dir1/fitxer_gran.txt"; done
for i in {1..100}; do echo "Línia $i d'un fitxer gran" >>
"$BASE_DIR/joc_proves/dir2/fitxer_gran.txt"; done

echo "Joc de proves creat correctament"
echo "Pots provar l'script amb els directoris joc_proves/dir1 i
joc_proves/dir2"
```

Contingut principal d'aquest script:

1. **Elimina** qualsevol versió prèvia de la carpeta `joc_proves` (amb `rm -rf`), per evitar conflictes.
2. **Determina** el directori base (`BASE_DIR`) localitzant la ubicació de l'script i anant "un nivell enrere" per situar la carpeta `joc_proves` al lloc esperat.
3. **Crea** l'arbre de directoris necessari:
    a. `dir1` i `dir2`, cadascun amb diversos subdirectoris (`subdir`, `ignorar`, `ignorar2`, etc.).
4. **Genera** un conjunt de **fitxers**:

a. Fitxers exactament iguals en ambdós directoris (`igual1.txt`, `igual2.txt`).
b. Fitxers amb **el mateix nom però contingut diferent** (`mateix_nom.txt`).
c. Fitxers **úniques** a cadascun dels directoris (`diferent1.txt`, `diferent2.txt`).
d. Fitxers **similars** (80%, 90% de semblança), per provar la funció de similitud.
e. Fitxers amb extensions (`.tmp`, `.bak`) per verificar l'opció d'ignorar-les.
f. Fitxers en subdirectoris per provar la comparació **recursiva**.
g. Fitxers amb permisos diferents (644 vs 755, 600 vs 700, etc.).

5. **Mostra** un missatge final: "Joc de proves creat correctament", indicant que ja pot provar-se l'script principal amb `joc_proves/dir1` i `joc_proves/dir2`.

2. Execució del script comparador

1) Fer executable l'script i executar-lo

Els passos són:

```
chmod +x scripts/crear_proves.sh
./scripts/crear_proves.sh
```

Això:

- **Assegura** que l'script tingui permisos d'execució (`chmod +x`).
- **Inicia** el procés de creació de totes les carpetes i fitxers en la ruta `joc_proves/dir1` i `joc_proves/dir2`.

2) Verificar el directori resultant

Un cop executat, es pot comprovar que ja existeix la carpeta `joc_proves` amb la següent estructura:

- `joc_proves/dir1/...`
- `joc_proves/dir2/...`
  - Amb tots els fitxers que s'han creat (igual1.txt, diferent1.txt, sub_igual.txt, etc.).

3) Invocar el comparador de directoris

Un cop **creat** l'entorn de proves, l'usuari pot fer:

```
chmod +x scripts/comparador_directoris.sh
./scripts/comparador_directoris.sh [opcions] <directori1> <directori2>
```

I aqui llavors aplicar les opcions que ofereix el comparador (com `-r`, `-p`, `-s <percent>`, etc.).

Així, **mitjançant** aquest script (`crear_proves.sh`), es garanteix un entorn **complet** i **preconfigurat** que cobreix gairebé totes les situacions que l'script comparador podria necessitar comprovar (fitxers idèntics, permissos diversos, extensions "temporal", subdirectoris ignorats, etc.).

Joc de proves i resultats

1. Comparació bàsica (només nivell superior)

`./scripts/comparador_directoris.sh joc_proves/dir1 joc_proves/dir2`

- **Què es fa?**
- L'script compara **exclusivament** els fitxers **al mateix nivell** dins `dir1` i `dir2` (sense entrar a subdirectoris, ja que no hi ha opció `-r`).
- **Què s'espera?**
  - Mostrar quins fitxers (a nivell principal) són **només** a `dir1` i quins són **només** a `dir2`.
  - Detectar si n'hi ha amb **el mateix nom** però contingut diferent i, en cas afirmatiu, mostrar les línies canviades (`diff`).
  - Calcular també la **similitud** (>= 90% per defecte) entre cada parella de fitxers en tots dos directoris.
- **Anàlisi de l'output** (parcial mostrat):
  - **Fitxers només a dir1**: hi surten, per exemple, `diferent1.txt` i `similar1.txt`.
  - **Fitxers només a dir2**: hi surten `diferent2.txt` i `similar2.txt`.
    - Això reflecteix que tenen **noms** diferents, de manera que l'script no els emparella com el mateix fitxer.
  - **Fitxer diferent**: `mateix_nom.txt`
    - Apareix un bloc `----- Diferències ... -----` amb la sortida de `diff`, indicant el canvi de contingut.
  - **Similitud**: Al final, apareixen missatges com:

```
El fitxer ".../permisos.txt" té una similitud del 100.00% amb
".../permisos.txt"
El fitxer ".../igual1.txt" té una similitud del 100.00% ...
...
```

Això confirma que l'script ha trobat **parelles** de fitxers amb el mateix nom i exactament el mateix contingut.

Aquest primer test confirma que la comparació bàsica funciona correctament.

2. Comparació recursiva

`./scripts/comparador_directoris.sh -r joc_proves/dir1 joc_proves/dir2`

- **Què es fa?**
- Aquí s'afegeix `-r`, de manera que es **busquen fitxers en tots els subdirectoris**.
- **Què s'espera?**
  - Ara apareixeran, a la secció "Fitxers només a `dir1` / `dir2`", també aquells fitxers que es trobin en subcarpetes que **no** tinguin correspondència en l'altra banda.
  - També si hi ha fitxers amb el mateix nom a `subdir/`, l'script mostraria si són diferents o iguals (100% similars).
- **Anàlisi de l'output** (parcial mostrat):
  - **Fitxers únics**: es veu que a `dir1` apareixen, per exemple, `subdir/unic_sub1.txt`, i a `dir2`, `subdir/unic_sub2.txt`.
  - **Fitxer diferent**: `subdir/sub_diff.txt`
    - `diff` mostra `'Contingut diferent subdir1'` vs `'Contingut diferent subdir2'`.
  - **Similitud**: Al final, s'observa que hi ha, per exemple,

```
El fitxer "/.../subdir/sub_igual.txt" té una similitud del 100.00%
amb "/.../subdir/sub_igual.txt"
```

Això corrobora que en **subdir** també hi ha fitxers idèntics.

Aquest test demostra la **funcionalitat recursiva**.

3. Comparació recursiva + comprovació de permisos

```
./scripts/comparador_directoris.sh -r -p joc_proves/dir1
joc_proves/dir2
```

- **Què es fa?**
- A més de la recursivitat, l'opció -p fa que l'script **compari els permisos** dels fitxers amb el mateix nom.
- **Què s'espera?**
  - En llistats "Fitxers només a X", haurien d'aparèixer els mateixos que a l'execució recursiva.
  - A més, si detecta que un fitxer a `dir1` té permisos diferents que el seu homòleg a dir2, l'script imprimeix:

```
Permisos diferents per fitxer1 i fitxer2:
fitxer1: 644
fitxer2: 755
```

- **Anàlisi de l'output** (parcial mostrat):

- Es repeteix la part de "Fitxers només a dir1/dir2" i "Fitxer diferent: ...".
- **Apareix** un bloc nou:

```
Permisos diferents per ../joc_proves/dir1/permisos.txt i
../joc_proves/dir2/permisos.txt:
../joc_proves/dir1/permisos.txt: 644
../joc_proves/dir2/permisos.txt: 755
```

Confirma que l'script ha trobat una discrepància en els modes octals.

Continua mostrant la **Similitud** al final (que pot ser 100% en el contingut, però s'informa igualment que els permisos són distints).

Així s'observa clarament la **funcionalitat addicional** per detectar permisos diferents fins i tot en fitxers idèntics pel que fa al contingut.

4. Ignorar fitxers temporals i de backup

`./scripts/comparador_directoris.sh -r -e tmp,bak joc_proves/dir1 joc_proves/dir2`

- **Què es fa?**
  S'estan **ignorant** tots els fitxers que acaben en `.tmp` i `.bak`.
- **Què s'espera?**
  No han d'aparèixer a llistes de "Fitxers només a ...", ni ser analitzats en la similitud, fitxers com `temporal.tmp` o `backup.bak`.
- **Anàlisi de l'output** (exemple):
  - Comprovar que no s'inclouen (o s'han reduït) els missatges de "El fitxer 'backup.bak' té una similitud del 100%...", etc.
  - Tot i que anteriorment apareixien, ara estan **exclosos** i no apareixen en les seccions de resultats.

Això confirma que la lògica de filtratge d'extensions funciona correctament.

5. Comparació recursiva ignorant el subdirectori "ignorar"

```
./scriptInicial.sh -r -d ignorar ../joc_proves/dir1
../joc_proves/dir2
```

Objectiu i funcionalitat

- **`-r`**: Cerca recursiva. Inclourà tots els subdirectoris, excepte els que ignorarem.
- **`-d ignorar`**: Exclou del llistat i comparació tot el que hi hagi dins el subdirectori anomenat "ignorar".

Sortida obtinguda (resum)

- Apareixen a **"Fitxers només a ..."**:
  - `diferent1.txt`, `similar1.txt`, `subdir/unic_sub1.txt` a `dir1`.
  - `diferent2.txt`, `similar2.txt`, `subdir/unic_sub2.txt` a `dir2`.
  - **No** surt `ignorar/ignorar_file.txt` (ni en dir1 ni en dir2) perquè l'opció `-d ignorar` el filtra.
- **Fitxer diferent**: `mateix_nom.txt` (amb la comparació de contingut que mostra `diff`).
- **Fitxer diferent**: `subdir/sub_diff.txt` (la subcarpeta "subdir" sí que s'analitza).
- **Similitud (>= 90%)**: mostra que `permisos.txt`, `igual1.txt`, `igual2.txt`, `backup.bak`, etc. són **100%** similars en contingut.

Anàlisi

- Això confirma que l'script **entra** a `subdir/`, però **no** processa res dins `ignorar/`.
- Per tant, `ignorar/ignorar_file.txt` no apareix.
- La resta de subdirectoris (com `subdir`) segueixen formant part de la comparació recursiva.

6. Comparació recursiva amb redirecció de sortida a un fitxer

```
./scriptInicial.sh -r -o resultats.txt ../joc_proves/dir1
../joc_proves/dir2
```

Objectiu i funcionalitat

- **`-r`**: activa la recerca en subdirectoris.
- **`-o resultats.txt`**: tota la **sortida** s'envia **dins** `resultats.txt` en lloc de mostrar-se per terminal.

Sortida obtinguda (exemple)

- Al fitxer `resultats.txt`, trobem seccions com:
  - "Fitxers només a ../joc_proves/dir1" → `diferent1.txt`, `ignor...`, etc.
  - "Fitxers només a ../joc_proves/dir2" → `diferent2.txt`, etc.
  - "Fitxer diferent: mateix_nom.txt" (amb la comparativa `diff`).
  - "Comparació de similitud entre tots els fitxers (>= 90%)": llistant els fitxers que tenen un 100% de coincidència de contingut.
- **A la terminal**, possiblement només veurem **informació mínima** (o res), perquè la major part està sent escrita en el fitxer de sortida.

Anàlisi

- Això **demostra** la capacitat de l'script per **desar** els resultats en un arxiu i, per tant, tenir un informe complet per revisió posterior.
- El contingut és el mateix que veuríem en pantalla, però a `resultats.txt`.

7. Combinació recursiva, amb permisos, ignorar extensions .tmp i .bak, ignorar el subdirectori "ignorar", llindar de similitud al 80% i sortida en fitxer

```
./scriptInicial.sh -r -p -e tmp,bak -d ignorar -s 80 -o
resultats.txt ../joc_proves/dir1 ../joc_proves/dir2
```

Objectiu i funcionalitat

- **`-r`**: fa la comparació recursiva.
- **`-p`**: comprova si hi ha permisos diferents.
- **`-e tmp,bak`**: exclou fitxers amb extensió `.tmp` o `.bak`.

- **-d ignorar**: ignora el subdirectori anomenat `ignorar`.
- **-s 80**: si un fitxer té **80%** o més de semblança, mostra un missatge de similitud (en lloc del 90% per defecte).
- **-o resultats.txt**: escriu tot al fitxer `resultats.txt`.

Sortida obtinguda (exemple)

- **Fitxers únics**:
  - `diferent1.txt`, `similar1.txt`, `subdir/unic_sub1.txt` a dir1,
  - `diferent2.txt`, `similar2.txt`, `subdir/unic_sub2.txt` a dir2.
  - No apareix cap `.tmp`, `.bak` (ni `ignorar/ignorar_file.txt`), perquè s'ha exclòs.
- **Fitxer diferent**: `mateix_nom.txt` (detalla el `diff`).
- **Permisos**:
  - Detecta permisos diferents en permisos.txt (644 vs 755) i ho mostra.
- **Similitud (>= 80%)**:
  - Indica 100% per a `igual1.txt`, `igual2.txt`, `sub_igual.txt`, etc.
  - (També consideraria fitxers amb 85% de semblança, per exemple, però en aquest cas, els fitxers de l'exemple arriben sovint al 100% o es queden per sota 80%.)

Anàlisi

- Aquesta és una prova **molt completa** perquè s'hi combinen tots els paràmetres.
- L'script filtra correctament .tmp i .bak, subdirectori "ignorar", i mostra la **comparació** tant de contingut com de **permisos**, i ajusta la **similitud** a un llindar de 80%.