ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

[DΣIM]

FONAMENTS DE

SISTEMES OPERATIUS

Curs 2024-25

Estudiant: Oupman Miralles

Professor/a:

ÍNDEX

PART 1: Execució de l'script inicial

En aquesta primera part l'objectiu és revisar i comentar el codi per tal d'entendre el seu funcionament i realitzar un joc de proves que demostri el correcte funcionament de l'script proporcionat.

Anàlisis de l'script

Aquest script compara dos directoris proporcionats per pàrametre, llista tots els fitxers presents en un directori que no estan continguts en l'altre, i detecta els fitxers que tenen el mateix nom però contingut diferent.

Primerament es comprova que es passin exactament 2 paràmetres (*#2*). En cas negatiu, es mostra un missatge d'error el qual es redirigeix a la *stderr (>&2)*, i acaba l'execució del programa.

```
1  #!/bin/bash
2
3  # ------------------------------------
4  # PART 1: Execucio de l'script inicial
5  # ------------------------------------
6
7  # Comprovar si es passen 2 params i existeixen
8  if [ "$#" -ne 2 ]; then
9      echo "Ús: $0 <directori1> <directori2>" >&2
10     exit 1
11 fi
```

A continuació s'aguarden els paràmetres en dues variables (DIR1 i DIR2) i es valida la seva existència (*! -d*). De nou, es mostra un missatge d'error el qual es redirigeix a la *stderr (>&2)*, i acaba l'execució del programa.

```
12 DIR1=$1
13 DIR2=$2
14 if [ ! -d "$DIR1" ] || [ ! -d "$DIR2" ]; then
15     echo "Un o ambdós directoris no existeixen." >&2
16     exit 1
17 fi
```

Després es llisten el contingu dels directoris. Per aconseguir-ho es fa ús de la commanda *comm*, que compara dos fitxers ordenats línia per línia[1]. Primer es llista tots els fitxers i directoris de *DIRx* i s'ordenen alfabèticament (*ls "$DIRx" | sort*). La sortida d'aquestes comandes es tracten com si fossin un fitxer gràcies a l'operador < (*redirecting input)*, i es redirigeix la entrada al *comm.* Finalment utilitzem les flags *-2* i *-3* per tal de suprimir la segona

[1] comm(1) – Linux man page

(fitxers únics a *DIR2*) i tercera (fitxers comuns) columna, mostrant així només la primera columna (futxers únics a *DIR1*). Fem el mateix per als fitxers de *DIR2* amb les flags *-1* i *-3*. El resultat final acaba sent els fitxers únics de cada directori.

```
18 # Llistar contingut dels directoris
19 # FLAGS:
20 #   -1: Suppress lines unique to file1.
21 #   -2: Suppress lines unique to file2.
22 #   -3: Suppress lines common to both files.
23 echo "Fitxers només a $DIR1:"
24 comm -23 <(ls "$DIR1" | sort) <(ls "$DIR2" | sort)
25 echo "Fitxers només a $DIR2:"
26 comm -13 <(ls "$DIR1" | sort) <(ls "$DIR2" | sort)
```

Per acabar tenim un bucle el qual s'encarrega de detectar els fitxers que tenen el mateix nom però contingut diferent. Itera per cada fitxer dins de DIR1, llavors comprova que aquell fitxer existeixi dins de DIR2, i si és així, utilitza la commanda *diff* per comparar els dos fitxers en mode *quiet* [2], el qual retorna un estat 0 si els fitxers són identics, i diferent de 0 si els fitxers són diferents. Redirigeix la sortida del *diff* al output */dev/null*, que descarta qualsevol informació que rep, ja que només ens interessa el estat que retorna *diff,* i no cap altra sortida de text. Amb aquest estat, si és diferent de 0 es mostrarà per pantalla el nom del fitxer que els dos directoris contenent amb el mateix nom però diferent contingut.

```
28 # Nomes mostra els fitxers amb el mateix nom
29 # i diferent contingut de DIR
30 for file in $(ls "$DIR1"); do
31     if [ -f "$DIR2/$file" ]; then
32         if ! diff -q "$DIR1/$file" "$DIR2/$file" >/dev/null; then
33             echo "Fitxer diferent: $file"
34         fi
35     fi
36 done
37 echo -e "\n"
```

Joc de proves

Es demana que el joc de proves generi dos directoris amb certs continguts:

- Fitxers iguals
- Fitxers diferents
- Subdirectoris
- Fitxers iguals i diferents en els subdirectoris

[2] diff(1) – Linux man page

El joc de proves que he realitzat s'encarrega de generar fitxers i directoris per després executar el script anterior, el qual he anomenat comparator.

En primer lloc m'asseguro de que existeixin els directoris de prova dir1 i dir2, si existeixen els elimino. Això està pensat per a que sempre es crein els fitxers prova i fer un funcionament automatitzat de l'script. En cas de que es volguéssin provar més fitxers, es podria modificar manualment els directoris i executar el comparator.sh per separat o inclouir nous fitxers al joc de proves. A continuació es llisten els directoris (comanda la qual donara error si no existeixen directoris, cosa que confirma la correcta eliminacó dels directoris de prova), i seguidament creo els dos directoris de prova nous i torno a llistar els directoris en aquesta ruta.

```
10 # Esborrar directoris anteriors
11 if [ -d dir1 ]; then
12     rm -r dir1
13 fi
14 if [ -d dir2 ]; then
15     rm -r dir2
16 fi
17 echo -e "\nDirectoris eliminats"
18 echo $(ls -d */)
19
20 # Crear directoris de proves
21 mkdir dir1 && mkdir dir2
22 echo -e " \nDirectoris dir1 dir2 creats:"
23 echo $(ls -d */)
24
```

En segon lloc, creo fitxers unics, iguals i diferents per a cada directori posant un contingut bàsic el qual redirecciono a cada fitxer amb el operador > (*redirecting output)*. També creo un subdirectori anomenat subdir, i li afegeixo d'igual manera fitxers unics, iguals i diferents. Finalment executo el comparator passant-li els directoris de prova per paràmetre.

```
25 # -------------------
26 # Fitxers
27 # -------------------
28
29 # Unics
30 echo "Only in dir1" > dir1/file1.txt
31 echo "Only in dir2" > dir2/file2.txt
32
33 # Iguals
34 echo "Equal file" > dir1/equal.txt
35 echo "Equal file" > dir2/equal.txt
36
37 # Diferents
38 echo "Different in dir1" > dir1/diff.txt
39 echo "Different in dir2" > dir2/diff.txt
```

```
41 # -------------------
42 # Subdirectoris
43 # -------------------
44 mkdir dir1/subdir
45 mkdir dir2/subdir
46
47 # Fitxers unics en dir1/subdir
48 echo "Unique in subdir1" > dir1/subdir/subfile1.txt
49 echo "Unique in subdir2" > dir1/subdir/subfile2.txt
50
51 # Fitxers iguals en els dos subdirs
52 echo "Equal in subdir" > dir1/subdir/subfile3.txt
53 echo "Equal in subdir" > dir2/subdir/subfile3.txt
54
55 # Fitxers diferents en dir1/subdir
56 echo "Different1 in subdir1" > dir1/subdir/subfile4.txt
57 echo "Different2 in subdir1" > dir1/subdir/subfile4.txt
58
59 # Fitxer unic en dir2/sudir
60 echo "Only in subdir2" > dir2/subdir/subfile5.txt
61
62
63 # Executar comparator.sh
64 echo -e "\nExecucio comparator.sh\n"
65
66 ./comparator.sh dir1 dir2
67
```

Conclusions

1. Eliminació i creació dels directoris de prova

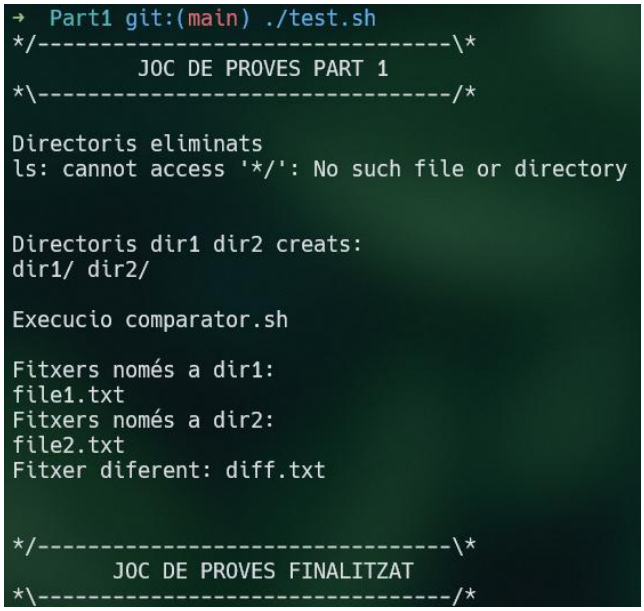
Es pot veure com es mostra l'error de la comanda ls, ja que estem intentar llistar tots els directoris, cosa que confirma l'eliminació dels directoris. Seguidament es confirma la creació dels directoris amb l'execució d'aquesta mateixa comanda.

2. Llistat dels fitxers

Observem com els fitxers únics de cada directori son llistats, mentre que els fitxers iguals no es mostren. D'igual manera podem observar que els fitxers amb mateix nom però diferent contingut es mostren a la secció de fitxers diferents.

3. Subdirectoris

Es pot observar com aquesta versió de l'script no és capaç de realitzar una busca recursiva, pel que tant els subdirectoris com els seus fitxers no es tenen en consideració. Això és degut a que la comanda ls nomes podrà llistar el contingut de l'arrel del directori.



PART 2: Ampliació de funcionalitats

La Part 2 té com a objectiu ampliar les funcionalitats del script inicial per oferir una comparació més avançada entre dos directoris. En aquesta versió s'han afegit diverses opcions configurables mitjançant paràmetres de línia de comandes, així com millores en la presentació dels resultats. Les funcionalitats adicionals que es demanaven i he realitzat son les següents:

- Comparació recursiva per incloure subdirectoris.
- Comparació avançada de fitxers amb càlcul de la similitud (en percentatge).
    - Mostrar el contingut de les linies diferents.
    - Ignorar espais o caràcters en blanc.
- Filtres per ignorar fitxers segons les seves extensions i subdirectoris.
- Comparació de permisos dels fitxers.
- Redirecció de la sortida a un fitxer, amb la possibilitat d'obrir-lo interactivament amb un editor seleccionat.
- Sortida amb codis de colors per diferenciar informació, errors, èxits i avisos.

Anàlisis de l'script

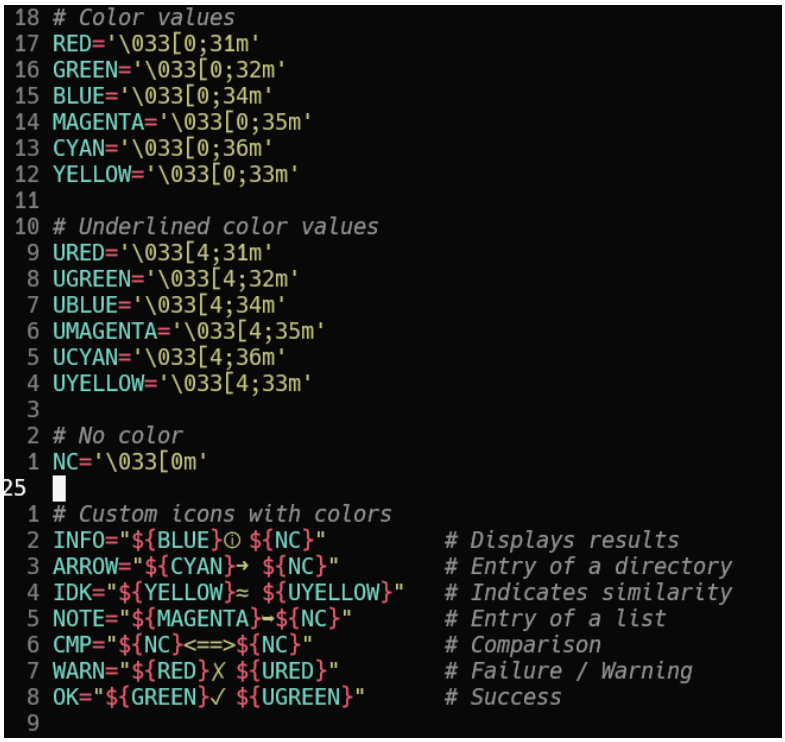
Inicialitzacions i definicions

L'script defineix constants amb codis ANSI per tal de donar colors i estils als missatgess de sortida, cosa que permet destacar títols, informació, resultats, avisos, èxits...

L'estructura d'aquests codis de colors és ben simple **\033[A;Bm**

- **\033**: tots aquests codis de colors han d'anar precedits del *escape code* per donar a coneixer que es tracta d'un codi ANSI. Aquest es pot representar com a \e (ASCII), \033 (Octal), \x1b (Hexa), els quals equivalent a 27 en decimal.
- **[** : marca el principi de la seqüència de control.
- **A**: es el numero que indica l'atribut del codi, el qual va del 0 al 9 i defineix l'estil del text com per exemple la negreta, cursiva, subrallat...
- **;** : separador entre l'atribut del text i el codi de color.
- **B:** es el numero que indica el codi de color. Especifica el color frontal (30-37) o del fons (40-47). Aquests 8 colors son per ordre: Negre, Vermell, Verd, Groc, Blau, Magenta, Cian, Blanc.

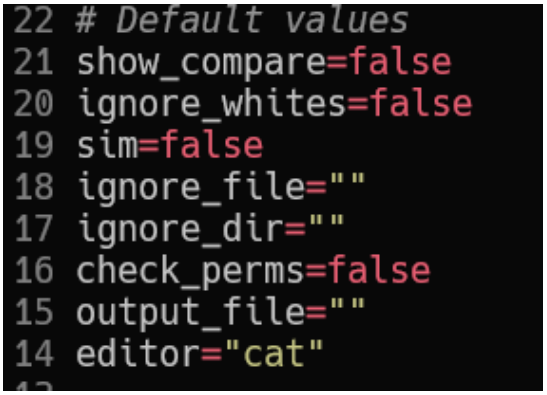
Tenint això en compte, he definit les constants de diferents colors i icones:



Tal i com es demanava a l'enunciat, s'han de configurar diferents paràmetres amb el getopts per poder utilizar o no certes funcionalitats de l'script a través de la línia de comandes.

Primer es declaren els valors per defecte de les variables que modificaran el getopts, els boleans s'inicialitzen a fals i els string com a cadenes de text buides.

- **show_compare (-c):** activa visualització de linies diferents entre fitxers.
- **ignore_whites (-w):** irgnore els espais i caràcters en blanc al comparar fitxers.
- **sim (-s):** realitza el càlcul de la similitud a nivell de caràcters entre fitxers.
- **ignore_file (-f):** ignora la llista d'extensions que es passa separada per comes.
- **ignore_dir (-d):** ignora el subdirectori especificat.
- **check_perms (-p):** activa la comparació de permisos entre fitxers.
- **output_file (-o):** es redirigeix les sortides en el fitxer especificat.



El funcionament del getopts es basa en un bucle que itera per a tota la cadena d'opcions que separen amb un colon **:** si aquestes esperan un argument. En aquest cas **c, w, s, p** no esperen aguments, mentre que **f, d, o** si, ja que son les que esperen un fitxer o subdirectori.

El bucle posa la variable **OPTIND** a 1 (index del següent argument a processar), i els arguments de cada opció és guarden en la variable **OPTARG** en cada iteració del bucle. Cada opció passa per un switch case, el qual realitza diferents operacions depenent de quina s'activi, i en cas de que no existeixi aquella opció, el getopts posa el **opt** a **?**, i el cas ***)** s'encarrega de gestionar l'error.

Una vegada s'han processat totes les opcions, el bucle acaba i és necessari fer un ***shift OPTIND-1*,** el qual elimina les opcions i els seus arguments per tal de deixar els arguments posicionals per a que puguin ser processats, en aquest cas **dir1** i **dir2**.

Cal destacar el cas **-o**, el qual requereix posar tots els colors com a cadenes buides per tal d'evitar que apareixin els codis de colors al fitxer de sortida.

```
# Parse getopt arguments
while getopts "cwsf:d:po:" opt; do
    case "$opt" in
        c)
            show_compare=true
            ;;
        w)
            ignore_whites=true
            ;;
        s)
            sim=true
            ;;
        f)
            ignore_file="$OPTARG"
            ;;
        d)
            ignore_dir="$OPTARG"
            ;;
        p)
            check_perms=true
            ;;
        o)
            # Disable colors to avoid file having color codes
            RED="" GREEN="" BLUE="" MAGENTA="" CYAN="" YELLOW="" NC=""
            URED="" UGREEN="" UBLUE="" UMAGENTA="" UCYAN="" UYELLOW=""
            INFO="ⓘ " ARROW="→ " IDK="≈ " NOTE="➥" CMP="<==>" WARN="✗ " OK="✓ "
            # Redirect output to the specified file
            output_file="$OPTARG"
            exec > "$output_file" 2>&1
            ;;
        *)
            echo "Invalid option: -${OPTARG}" >&2
            exit 1
            ;;
    esac
done
shift $((OPTIND-1)) # Remove opts args to access positional args
```

De la mateixa manera que en l'script inicial, es validen els arguments posicionals **dir1** i **dir2**.

```
15 # Comprovar si es passen 2 params i existeixen
16 if [ "$#" -ne 2 ]; then
17     echo "Ús: $0 [-f extensio1,extensio2,...] [-d subdirectori] <directori1> <directori2>"
18
19     exit 1
20 fi
21 DIR1=$1
22 DIR2=$2
23 if [ ! -d "$DIR1" ] || [ ! -d "$DIR2" ]; then
24     echo "Un o ambdós directoris no existeixen." >&2
25     exit 1
26 fi
```

Comparació recursiva

Per aconseguir que la comparació de fitxers es faci de forma recursiva i identificar els fitxers unics de cada directori he utilitzat la comanda ***find*** amb les opcions:

- **-type f**: per només comparar fitxers i no directoris.
- **-printf "%P\n"**: per mostrar el path relatiu del directori.

I el funcionament de *process substitiuion* i les flags del ***comm*** funcionen igual que en l'script inicial. El resultat li canvio el format perquè es vegi millor amb la comanda **sed**, la qual li especifico que per cada principi de linia (**^**) faci una tabulació (**\t**) i posi la icona de ARROW amb l'us d'un **printf**, ja que sed per defecte no accepta el *escape*.

```
6 echo -e "Fitxers només a ${CYAN}$DIR1${NC}:\n"
5 comm -23 <(find "$DIR1" -type f -printf "%P\n" | sort) <(find "$DIR2" -type f -printf "%P\n" | sort) | sed "s|^|\t`printf "${ARROW}"`|"
4 echo -e "\nFitxers només a ${CYAN}$DIR2${NC}:\n"
3 comm -13 <(find "$DIR1" -type f -printf "%P\n" | sort) <(find "$DIR2" -type f -printf "%P\n" | sort) | sed "s|^|\t `printf "${ARROW}"`|"
```

Comparació avançada de fitxers

La funció de **advanced_comparison** rep els dos fitxers a comparar per paràmetre, i les guarda com a variables locals.

Per obtenir les diferències entre linies utilitzo la comanda **diff** amb diferents flags depenent de si l'opció **-w** està activa o no.

+ S'utlitzen les flags -B -w -u0, les quals ignoren les linies blanques (B), espais en blanc (w) i mostra el resultat unificat (u) només mostran les linies diferents (0).
– Només s'utliza la flag -u0 per mostrar el resultat unificat de les línies diferents.

Es fa un grep invertint la condició per tal de no mostrar les línies especials que comencen amb @@, ja que només ens interessen les linies que comencen amb -/+, que son les que mostren les diferencies, amb el --- indicant el primer arxiu i +++ el segon.

Si l'opció **-c** està activa i les linies diferents no son zero, aleshores es mostren les linies diferents obtingudes anteriorment amb el **diff** i el **grep**.

Si l'opció **-s** està activa es crida la funció de calcular similitut passant per paràmetre els 2 fixters a comparar, i si la similitut es superior al 90% es mostren els paths absoluts dels fixters per pantalla, juntament amb el % de similitud indeferentment d'aquest.

Finalment es mostra una linia amb el resultat de la comparació avançada.

- **Fitxers iguals:** si les linies diferents son 0.
- **Fitxers molt semblants:** si la similitut dels fitxers es superior al 90%.
- **Fitxers diferetns:** si la similitut dels fitxers es inferior al 90%.

```
29 # 2. Advanced comparison
28 advanced_comparison() {
27     local file1="$1"
26     local file2="$2"
25
24     # Get the total number of lines in each file
23     if $ignore_whites; then
22         local diffed=$(diff -B -w -u0 "$file1" "$file2" | grep -v '^@@')
21     else
20         local diffed=$(diff -u0 "$file1" "$file2" | grep -v '^@@')
19     fi
18
17     # If flag is set and files are not the same, show the differences
16     if $show_compare && [[ -n "$diffed" ]]; then
15         echo -e "\n${INFO}Linies diferents:\n" | sed "s|^|\t|"
14         echo "$diffed" | sed "s|^|\t\t|"
13         echo ""
12     fi
11
10     # Return the absolute path if the similarity is greater than 90%
 9     if $sim; then
 8         local similarity=$(calculate_similarity "$file1" "$file2")
 7         echo -e "${INFO}Els fitxers tenen un ${MAGENTA}$similarity%${NC} de similitud\n" | sed "s|^|\t|"
 6         if [ "$similarity" -ge 90 ]; then
 5             # Print the absolute path of the files
 4             echo -e "${ARROW}${BLUE} $(realpath -e "$file1")${NC}" | sed "s|^|\t\t|"
 3             echo -e "${ARROW}${BLUE} $(realpath -e "$file2")${NC}\n" | sed "s|^|\t\t|"
 2         fi
 1     fi
177
 1     # Print the result
 2     if [ -z "$diffed" ]; then
 3         echo -e "${OK}Els fitxers son iguals${NC}" | sed "s|^|\t|"
 4     else
 5         if $sim && [[ "$similarity" -ge 90 ]]; then
 6             echo -e "${IDK}Els fitxers son molt semblants${NC}" | sed "s|^|\t|"
 7         else
 8             echo -e "${WARN}Els fitxers son diferents${NC}" | sed "s|^|\t|"
 9         fi
10     fi
11 }
NORMAL  main
```

Ignorar certs fitxers

Per ignorar certs fitxers o subdirectoris tenim les flags -f, -d. La -f espera una cadena de extensions de fitxers a ignorar (inclouent el punt) i separades per una coma, i la flag -d espera el nom del subdirectori a ignorar.

Si es proporciona una cadena d'extensions, aquesta es converteix en un array anomenada extensions, la qual guarda cada extensió (.ext) en cada posició del array, que s'aconseguiex indicant que el separador IFS es la coma.

```
13 # Convert the comma-separated string into an array
12 if [ -n "$ignore_file" ]; then
11     IFS=',' read -r -a extensions <<< "$ignore_file"
10     echo -e "\nComparacio avancada de fitxers ignorant (${YELLOW}${extensions[@]}${NC}):\n"
 9 else
 8     echo -e "\nComparacio avancada de fitxers:\n"
 7 fi
 6
```

El bucle principal consta de dues parts. La primera recorre el dir1 recursivament, i per a cada fitxer imprimeix la seva ruta relativa. La segona agafa la sortida del **find** anterior i la guarda a la variable relpath sense interpretar el *escape*, i el IFS= garanteix que no es tinguin en compte els espais com a separadors.

Si el path relatiu del dir1 es un fitxer existen dins del dir2 (es a dir que tenen el mateix nom), comprovarem els filteres que s'hagin d'aplicar per tal de realitzar la comparació avançada.

Primer comprovem si s'ha prorporcionat una cadena de caràcters per ignorar un directori, si és així comprovarem que el fitxer que volem analitzar no es trobi dins del subdirectori especificat. Si aquesta condicio es compleix el boolea **skip** es posarà a cert i no es realitzarà la comparació i passarem al següent fitxer.

A continuació ess comprova si s'ha proporcionat una cadena amb les extensions de fixters, de ser el cas es comprova per cada extensió continguda al array d'extensions si el fitxer que es vol comparar té la mateixa extensió que alguna dins de l'array. En cas afirmatiu, ignorariem aquell fitxer saltant la comprovació avançada.

Posteriorment, comprovem si hem de fer o no la comprovació mirant el valor del booleà **skip**. En cas que s'hagi de fer, comprovem que la sortida de la comanda **diff** en mode *quiet* si els fitxers son iguals o diferents. En cas de ser diferents s'invoca la funció **advanced_comparison** amb els 2 fitxers com argument, i en cas contrari es mostra per pantalla que els fitxers son iguals i no es realitza la comparació.

Finalment, si no hem de ignorar els fitxers, comprovem si la hem de comparar permisos, en cas afirmatiu s'invoca la funció **perms_comparison** amb els 2 fitxers com a paràmetre.

```
 6
 5 # Apply filters to the files
 4 find "$DIR1" -type f -printf "%P\n" | while IFS= read -r relpath; do
 3     if [ -f "$DIR2/$relpath" ]; then
 2         skip=false
 1
227         # Check if the subdirectory is in the ignore list
 1         if [ -n "$ignore_dir" ]; then
 2             if [[ "$relpath" == "$ignore_dir"* ]]; then
 3                 skip=true
 4             fi
 5         fi
 6
```

```
        # Check if the file extension is in the ignore list
        if [ -n "$ignore_file" ]; then
            for ext in "${extensions[@]}"; do
                if [[ "$relpath" == *"$ext" ]]; then
                    skip=true
                    break
                fi
            done
        fi
        # Compare the files if they are not skipped
        if ! $skip; then
            echo ""
            echo -e " ${NOTE} Comparant: ${CYAN}$DIR1/$relpath ${CMP} ${CYAN}$DIR2/$relpath${NC}\n"
            if ! diff -q "$DIR1/$relpath" "$DIR2/$relpath" >/dev/null; then
                advanced_comparison "$DIR1/$relpath" "$DIR2/$relpath"
            else
                echo -e "${OK}Els fitxers son iguals${NC}" | sed "s|^|\t|"
            fi

            # Check the permissions of the files
            if $check_perms; then
                perms_comparison "$DIR1/$relpath" "$DIR2/$relpath"
            fi
        fi
    fi
done
```

Comprovació de permisos

Per a tal de verificar els permisos dels fitxers a comparar tenim la flag -p, que com explicat anteriorment, si està activa la flag es crida la funció, i aquesta rep per paràmetre els 2 fitxers a comparar. Amb la comanda **stat**[3] es poden veure els permisos dels fitxers, i amb el flag **-c** es pot especificar el format. Per visualitzar els permisos en format octal s'utilitza %a, i per format lèxic %A.

Es guarden els permisos en octal de cada fitxer en una variable i es comparen, i aleshores es mostra per pantalla el resultat de la comparació mostrant el path relatiu del fitxer i els permisos en format lèxic i octal. Per a mostrar els missatges es fa ús dels colors i icones configurats anteriorment. Una creu i color vermell per quan son diferents, i un tick i color verd quan son iguals.

```
perms_comparison() {
    local file1="$1"
    local file2="$2"

    # Check the permissions of the files
    local perms1=$(stat -c "%a" "$file1")
    local perms2=$(stat -c "%a" "$file2")
    echo ""
    if [ "$perms1" != "$perms2" ]; then
        echo -e "${WARN}Els permisos son diferents${NC}\n" | sed "s|^|\t|"
        echo -e "${ARROW} "$file1"" | sed "s|^|\t\t|"
        echo -e "${INFO}${YELLOW} $(stat -c "%A" "$file1") ($perms1)${NC}" | sed "s|^|\t\t\t|"
        echo -e "${ARROW} "$file2"" | sed "s|^|\t\t|"
        echo -e "${INFO}${YELLOW} $(stat -c "%A" "$file2") ($perms2)${NC}" | sed "s|^|\t\t\t|"
    else
        echo -e "${OK}Els permisos son iguals${NC}\n" | sed "s|^|\t|"
        echo -e "${ARROW} "$file1"" | sed "s|^|\t\t|"
        echo -e "${INFO}${YELLOW} $(stat -c "%A" "$file1") ($perms1)${NC}" | sed "s|^|\t\t\t|"
        echo ""
    fi
}
```

[3] stat(1) – Linux man page

Registre en un fitxer

Com bé s'ha mencionat a l'apartat del getopts, per poder guardar els resultats de l'script en fitxer de registre és necessari posar les constants dels colors com a cadenes buides per evitar que el fitxer guardi els codis dels colors, ja que aquests només es poden visualitzar per la sortida del terminal.

He obtat per simplement redirigir totes les sortides dels **echo** al fitxer de sortida amb nom especificat per argument, que s'aconsegueix amb 2>&1 que son les sortides **stdout** i **stderr.**

```
38          o)
37              # Disable colors to avoid file having color codes
36              RED="" GREEN="" BLUE="" MAGENTA="" CYAN="" YELLOW="" NC=""
35              URED="" UGREEN="" UBLUE="" UMAGENTA="" UCYAN="" UYELLOW=""
34              INFO="ⓘ " ARROW="➜ " IDK="≈ " NOTE="➥" CMP="<==>" WARN="✗ " OK="✓ "
33              # Redirect output to the specified file
32              output_file="$OPTARG"
31              exec > "$output_file" 2>&1
30              ;;
```

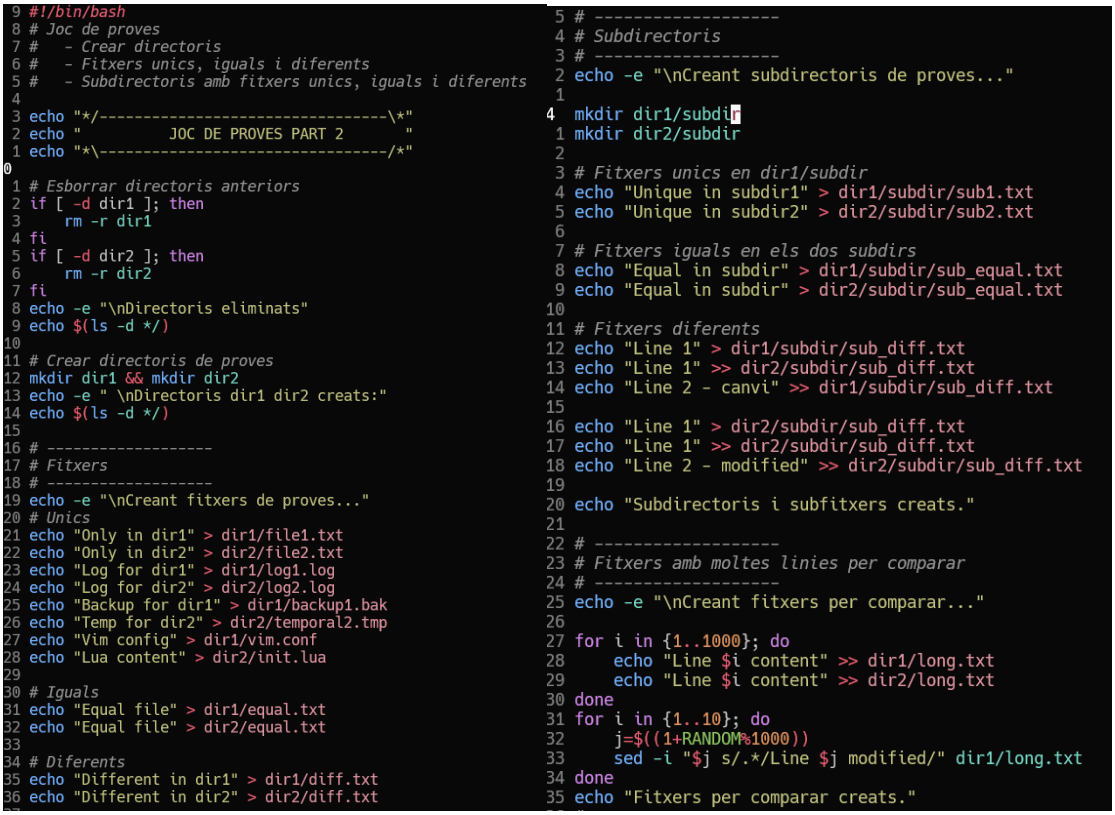
Adicionalment, per idea de disseny he incorporat una opció per fer que l'usuari pugui visualitzar el fitxer creat amb algun editor de text. Dels quals es troben vim, neovim, nano i cat (per veure el contingut del fitxer en el terminal). Com que totes les sortides son redirigides al fitxer, per poder mostrar aquests missatges al terminal les he direccionat a la sortida **/dev/tty** que representa el terminal associat al process actual, garantin així que tant el missatge com la lectura de la resposta vagin i provinguin del terminal.

El funcionament és tant basic com agafar la resposta, i si es afirmativa, es llegeix l'opció del terminal, i amb un switch case es construeix la comanda d'execució per obrir el fitxer amb l'editor seleccionat.

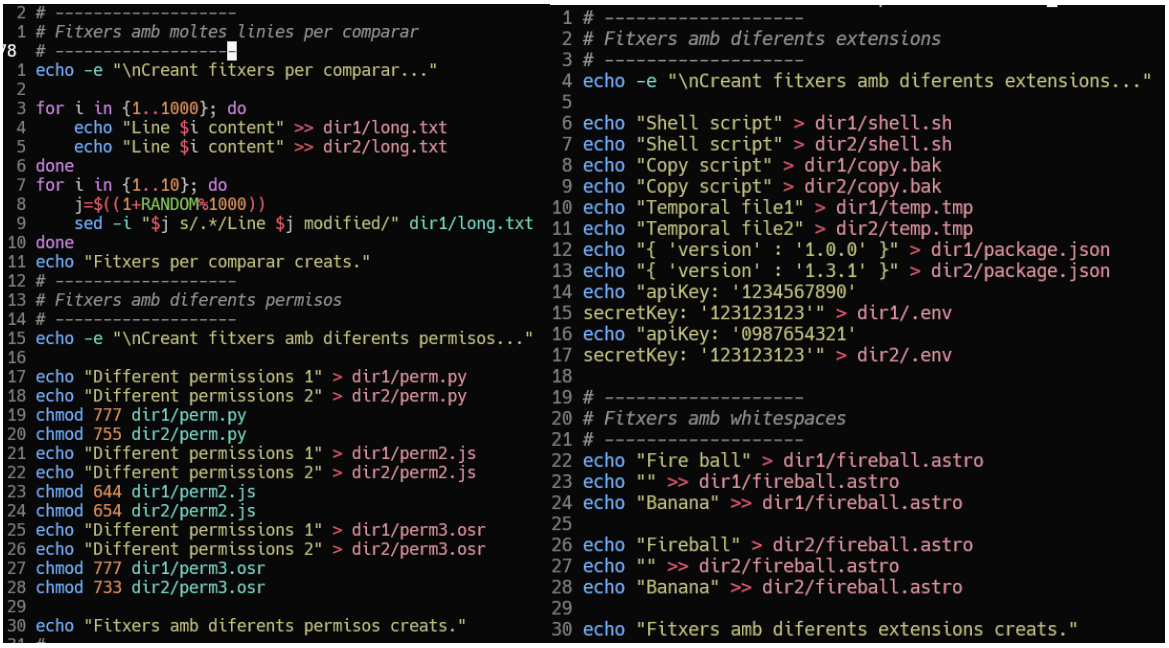
```
39 # Open output file
40 # dev/tty is used to read input from the terminal
41 if [ -n "$output_file" ]; then
42     echo "Resultats guardats a $output_file" > /dev/tty
43     echo "Vols veure els resultats? (y/n)" > /dev/tty
44     read -r res < /dev/tty
45     if [ "$res" == "y" ]; then
46         echo "Com el vols veure?"> /dev/tty
47         echo "vim[1], neovim[2], nano[3], cat[4]: " > /dev/tty
48         read -r option < /dev/tty
49         case $option in
50             1) editor="vim";;
51             2) editor="nvim";;
52             3) editor="nano";;
53             4) editor="cat";;
54             *) echo "Opció no vàlida. S'obrirà amb cat." > /dev/tty;;
55         esac
56         echo "Obrint amb $editor..." > /dev/tty
57         $editor $output_file > /dev/tty
58     fi
59 fi
```

Joc de proves

El joc de proves segueix l'estructura del joc de proves de l'script 1. Creo fitxers unics, iguals, diferents i amb diferents extensions per a fer diferents comparacions.



```bash
#!/bin/bash
# Joc de proves
#   - Crear directoris
#   - Fitxers unics, iguals i diferents
#   - Subdirectoris amb fitxers unics, iguals i diferents

echo "*/--------------------------------\*"
echo "          JOC DE PROVES PART 2       "
echo "*\--------------------------------/*"

# Esborrar directoris anteriors
if [ -d dir1 ]; then
    rm -r dir1
fi
if [ -d dir2 ]; then
    rm -r dir2
fi
echo -e "\nDirectoris eliminats"
echo $(ls -d */)

# Crear directoris de proves
mkdir dir1 && mkdir dir2
echo -e " \nDirectoris dir1 dir2 creats:"
echo $(ls -d */)

# -------------------
# Fitxers
# -------------------
echo -e "\nCreant fitxers de proves..."
# Unics
echo "Only in dir1" > dir1/file1.txt
echo "Only in dir2" > dir2/file2.txt
echo "Log for dir1" > dir1/log1.log
echo "Log for dir2" > dir2/log2.log
echo "Backup for dir1" > dir1/backup1.bak
echo "Temp for dir2" > dir2/temporal2.tmp
echo "Vim config" > dir1/vim.conf
echo "Lua content" > dir2/init.lua

# Iguals
echo "Equal file" > dir1/equal.txt
echo "Equal file" > dir2/equal.txt

# Diferents
echo "Different in dir1" > dir1/diff.txt
echo "Different in dir2" > dir2/diff.txt

# -------------------
# Subdirectoris
# -------------------
echo -e "\nCreant subdirectoris de proves..."

mkdir dir1/subdir
mkdir dir2/subdir

# Fitxers unics en dir1/subdir
echo "Unique in subdir1" > dir1/subdir/sub1.txt
echo "Unique in subdir2" > dir2/subdir/sub2.txt

# Fitxers iguals en els dos subdirs
echo "Equal in subdir" > dir1/subdir/sub_equal.txt
echo "Equal in subdir" > dir2/subdir/sub_equal.txt

# Fitxers diferents
echo "Line 1" > dir1/subdir/sub_diff.txt
echo "Line 1" >> dir2/subdir/sub_diff.txt
echo "Line 2 - canvi" >> dir1/subdir/sub_diff.txt

echo "Line 1" > dir2/subdir/sub_diff.txt
echo "Line 1" >> dir2/subdir/sub_diff.txt
echo "Line 2 - modified" >> dir2/subdir/sub_diff.txt

echo "Subdirectoris i subfitxers creats."

# -------------------
# Fitxers amb moltes linies per comparar
# -------------------
echo -e "\nCreant fitxers per comparar..."

for i in {1..1000}; do
    echo "Line $i content" >> dir1/long.txt
    echo "Line $i content" >> dir2/long.txt
done
for i in {1..10}; do
    j=$((1+RANDOM%1000))
    sed -i "$j s/.*/Line $j modified/" dir1/long.txt
done
echo "Fitxers per comparar creats."
```



```bash
# -------------------
# Fitxers amb moltes linies per comparar
# -------------------
echo -e "\nCreant fitxers per comparar..."

for i in {1..1000}; do
    echo "Line $i content" >> dir1/long.txt
    echo "Line $i content" >> dir2/long.txt
done
for i in {1..10}; do
    j=$((1+RANDOM%1000))
    sed -i "$j s/.*/Line $j modified/" dir1/long.txt
done
echo "Fitxers per comparar creats."
# -------------------
# Fitxers amb diferents permisos
# -------------------
echo -e "\nCreant fitxers amb diferents permisos..."

echo "Different permissions 1" > dir1/perm.py
echo "Different permissions 2" > dir2/perm.py
chmod 777 dir1/perm.py
chmod 755 dir2/perm.py
echo "Different permissions 1" > dir1/perm2.js
echo "Different permissions 2" > dir2/perm2.js
chmod 644 dir1/perm2.js
chmod 654 dir2/perm2.js
echo "Different permissions 1" > dir1/perm3.osr
echo "Different permissions 2" > dir2/perm3.osr
chmod 777 dir1/perm3.osr
chmod 733 dir2/perm3.osr

echo "Fitxers amb diferents permisos creats."
# -------------------
# Fitxers amb diferents extensions
# -------------------
echo -e "\nCreant fitxers amb diferents extensions..."

echo "Shell script" > dir1/shell.sh
echo "Shell script" > dir2/shell.sh
echo "Copy script" > dir1/copy.bak
echo "Copy script" > dir2/copy.bak
echo "Temporal file1" > dir1/temp.tmp
echo "Temporal file2" > dir2/temp.tmp
echo "{ 'version' : '1.0.0' }" > dir1/package.json
echo "{ 'version' : '1.3.1' }" > dir2/package.json
echo "apiKey: '1234567890'
secretKey: '123123123'" > dir1/.env
echo "apiKey: '0987654321'
secretKey: '123123123'" > dir2/.env

# -------------------
# Fitxers amb whitespaces
# -------------------
echo "Fire ball" > dir1/fireball.astro
echo "" >> dir1/fireball.astro
echo "Banana" >> dir1/fireball.astro

echo "Fireball" > dir2/fireball.astro
echo "" >> dir2/fireball.astro
echo "Banana" >> dir2/fireball.astro

echo "Fitxers amb diferents extensions creats."
```

```
# --------------------
# Fitxers d'un directory a ignorar
# --------------------
echo -e "\nCreant fitxers d'un subdirectori a ignorar..."

mkdir dir1/ignore && mkdir dir2/ignore
echo "margin: 4px;
padding: 2px 4px;
border: none;
color: white;" > dir1/ignore/styles.css
echo "margin: 0px;
padding: 2px 4px;
border: bold;
color: red;" > dir2/ignore/styles.css

echo "Fitxers d'un subdirectori a ignorar creats."
```

La gran diferència és el menu de proves, el qual permet a l'usuari probar el funcionament de totes les flags per separat i conjuntament, seleccionant del 0 al 9, A per veure-les totes alhora i X per sortir.

```
echo "Fitxers d'un subdirectori a ignorar creats."
# --------------------
# Executar comparator.sh
# --------------------
echo -e "\nExecutant comparator.sh amb diferents opcions..."
while true; do
    echo ""
    echo "--------------------------------------"
    echo "--------------------------------------"
    echo "Selecciona el cas de prova:"
    echo "    [0] Sense parametres"
    echo "    [1] Amb parametres incorrectes (directori inexistent)"
    echo "    [2] Amb parametres correctes (dir1 i dir2)"
    echo "    [3] Amb flag -c (mostrant comparacio de linies)"
    echo "    [4] Amb flag -w (ignorant whitespaces)"
    echo "    [5] Amb flag -s (calculant similaritat)"
    echo "    [6] Amb flag -f (ignorant .bak i .txt)"
    echo "    [7] Amb flag -d (ignorant dirX/ignore)"
    echo "    [8] Amb flag -p (mostrant permisos)"
    echo "    [9] Amb flag -o (guardant sortida a register.log)"
    echo "    [A] Amb tots els flags (-c, -w, -s, -f, -d, -p, -o)"
    echo "    [X] Sortir"
    echo "--------------------------------------"
    echo "--------------------------------------"
    echo ""
```

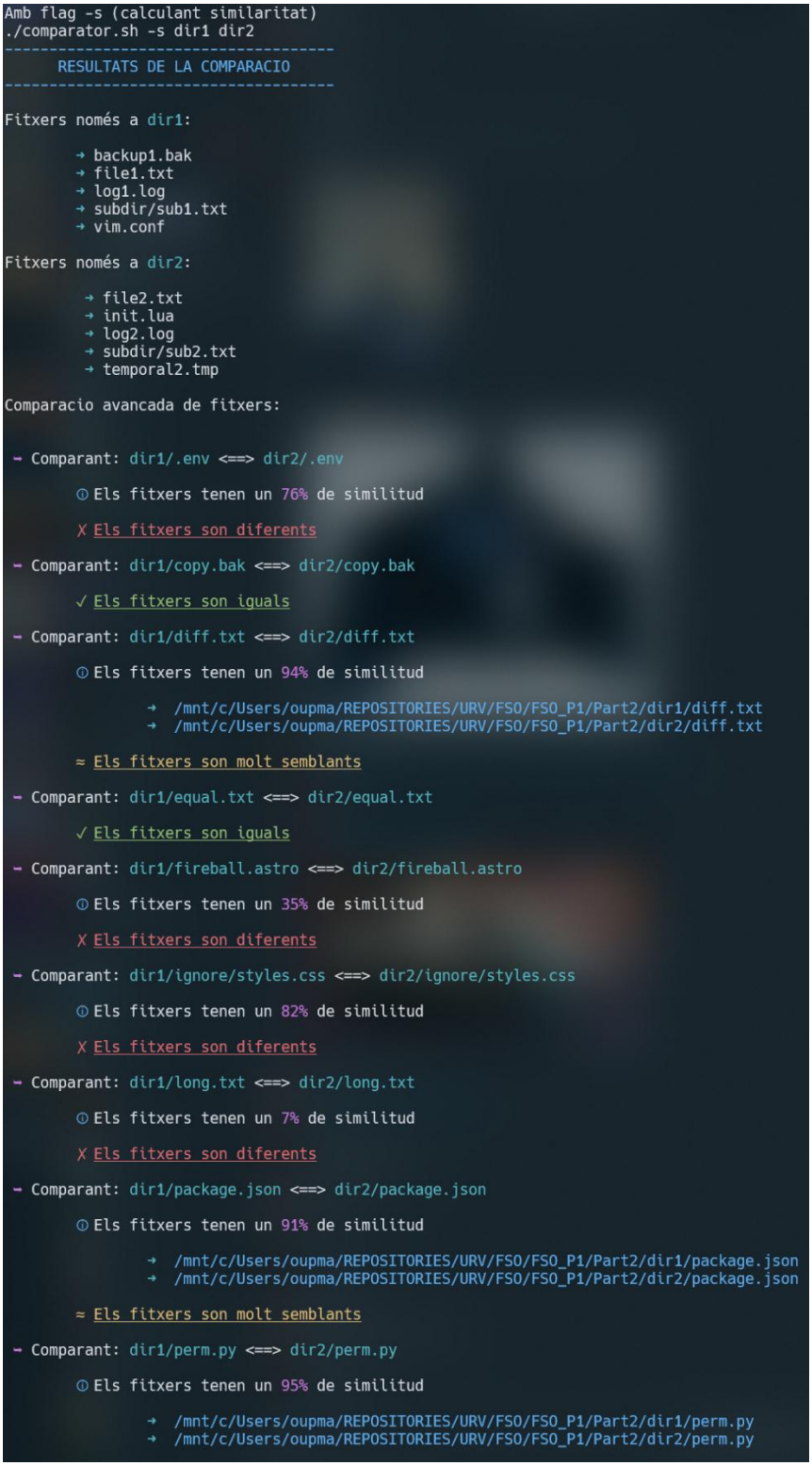
```
    read -p "Tria una opcio [0-9]: " option

    case $option in
        0)
            echo -e "\nSense parametres"
            echo "./comparator.sh"
            ./comparator.sh
            ;;
        1)
            echo -e "\nAmb parametres incorrectes"
            echo "./comparator.sh dir1 inexistent"
            ./comparator.sh dir1 inexistent
            ;;
        2)
            echo -e "\nAmb parametres correctes"
            echo "./comparator.sh dir1 dir2"
            ./comparator.sh dir1 dir2
            ;;
        3)
            echo -e "\nAmb flag -c (mostrant comparacio de linies)"
            echo "./comparator.sh -c dir1 dir2"
            ./comparator.sh -c dir1 dir2
            ;;
        4)
            echo -e "\nAmb flag -w (ignorant whitespaces)"
            echo -e "\nIgnorant tots els fitxers menys el de prova de whitespaces"
            echo "./comparator.sh -w -c -f .txt,.log,.bak,.tmp,.conf,.lua,.py,.osr,.js,.sh,.json,.env,.css dir1 dir2"
            ./comparator.sh -w -c -f .txt,.log,.bak,.tmp,.conf,.lua,.py,.osr,.js,.sh,.json,.env,.css dir1 dir2
            echo -e "\nSense flag -w (comparant whitespaces)"
            echo "./comparator.sh -c -f .txt,.log,.bak,.tmp,.conf,.lua,.py,.osr,.js,.sh,.json,.env,.css dir1 dir2"
            ./comparator.sh -c -f .txt,.log,.bak,.tmp,.conf,.lua,.py,.osr,.js,.sh,.json,.env,.css dir1 dir2
            ;;
```

```
+   6         5)
+   5             echo -e "\nAmb flag -s (calculant similaritat)"
+   4             echo "./comparator.sh -s dir1 dir2"
+   3             ./comparator.sh -s dir1 dir2
+   2             ;;
+   1         6)
  219             echo -e "\nAmb flag -f (ignorant .bak i .txt)"
+   1             echo "./comparator.sh -f .bak,.txt dir1 dir2"
    2             ./comparator.sh -f .bak,.txt dir1 dir2
    3             ;;
~   4         7)
    5             echo -e "\nAmb flag -d (ignorant dirX/ignore)"
+   6             echo "./comparator.sh -d ignore dir1 dir2"
    7             ./comparator.sh -d ignore dir1 dir2
    8             ;;
~   9         8)
   10             echo -e "\nAmb flag -p (mostrant permisos)"
+  11             echo "./comparator.sh -p dir1 dir2"
   12             ./comparator.sh -p dir1 dir2
   13             ;;
~  14         9)
~  15             echo -e "\nAmb flag -o (guardant sortida a register.log)"
~  16             echo "./comparator.sh -o register.log dir1 dir2"
+  17             ./comparator.sh -o register.log dir1 dir2
   18             ;;
~  19         A)
~  20             echo -e "\nAmb tots els flags (-c, -w, -s, -f, -d, -p, -o)"
+  21             echo "./comparator.sh -c -w -s -f .bak,.txt -d ignore -p -o output.log dir1 dir2"
   22             ./comparator.sh -c -f .bak,.txt -d ignore -p -o output.log dir1 dir2
   23             ;;
~  24         X)
   25             break
   26             ;;
   27         *)
   28             echo "Opcio incorrecta, torna a provar"
   29             ;;
   30     esac
   31 done
   32
   33 echo "*/--------------------------------\*"
   34 echo "        JOC DE PROVES FINALITZAT     "
   35 echo "*\--------------------------------/*"
```

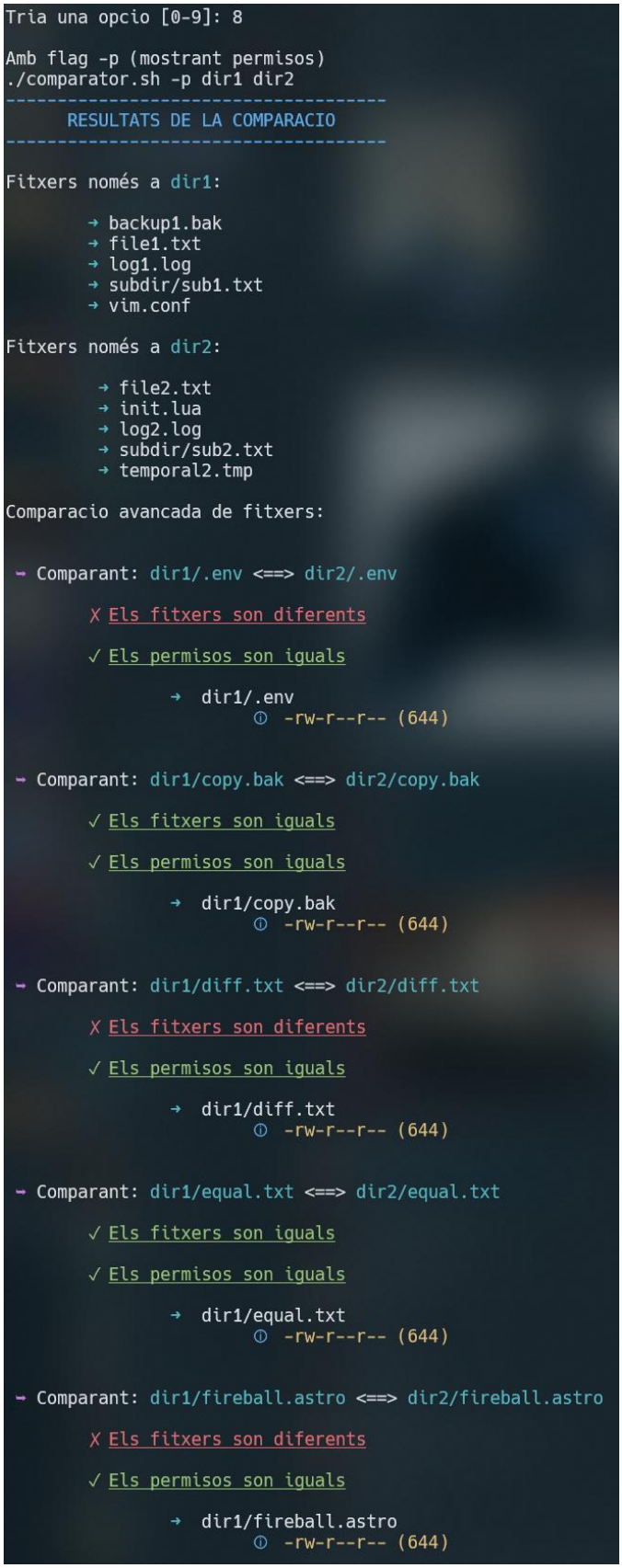
Conclusions

Els diferents casos han pogut corroborar el correcte funcionament del programa tal i com es demanava en l'enunciat.

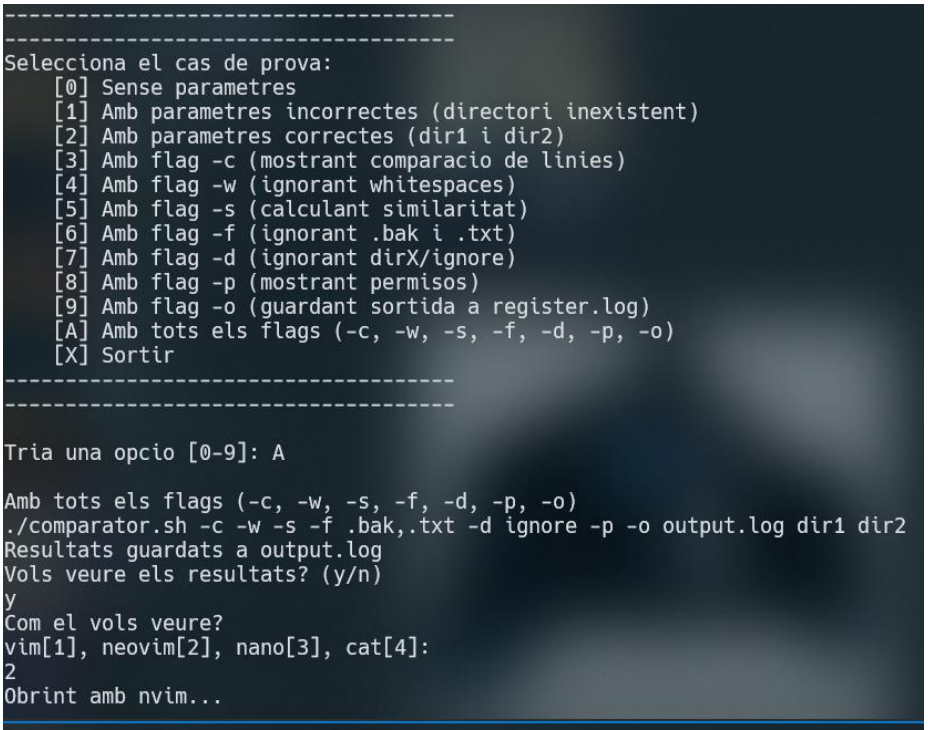
- Prova num 5:



- Prova num 8:


Tria una opcio [0-9]: 8
Amb flag -p (mostrant permisos)
./comparator.sh -p dir1 dir2
RESULTATS DE LA COMPARACIO
Fitxers només a dir1:
backup1.bak
file1.txt
log1.log
subdir/sub1.txt
vim.conf
Fitxers només a dir2:
file2.txt
init.lua
log2.log
subdir/sub2.txt
temporal2.tmp
Comparacio avancada de fitxers:
Comparant: dir1/.env <==> dir2/.env
Els fitxers son diferents
Els permisos son iguals
dir1/.env
-rw-r--r-- (644)
Comparant: dir1/copy.bak <==> dir2/copy.bak
Els fitxers son iguals
Els permisos son iguals
dir1/copy.bak
-rw-r--r-- (644)
Comparant: dir1/diff.txt <==> dir2/diff.txt
Els fitxers son diferents
Els permisos son iguals
dir1/diff.txt
-rw-r--r-- (644)
Comparant: dir1/equal.txt <==> dir2/equal.txt
Els fitxers son iguals
Els permisos son iguals
dir1/equal.txt
-rw-r--r-- (644)
Comparant: dir1/fireball.astro <==> dir2/fireball.astro
Els fitxers son diferents
Els permisos son iguals
dir1/fireball.astro
-rw-r--r-- (644)

- Prova amb totes les flags (A):



Execució

Per executar els scripts és tant senzill com fer **cd** a la Part que es vulgui testejar, donar permisos d'execució amb **chmod +x** i executar el **./test.sh.** També es pot utilitzar unicament **comparator.sh** si es vol executar de forma manual, pero el **test.sh** de les dues parts ja s'encarrega d'executar-los amb les diverses proves.