ERASMUS, UNIVERSITY OF WEST ATTICA

Report

Discipline: **Operating Systems**

Student: Istrati Daniel

Greece, Athens – 2025

Task

A. Write a script called **searching** that (a) accepts two integers as arguments and (b) asks the user for a directory name, and then displays the following (for items 1-3 use the **find** command, for items 4-5 use **ls** and **grep** commands in pipeline):

1. The files of the tree of the given directory with permissions equal (as an octal equivalent) to the first number (argument).
2. The files of the tree of the given directory that changed contents during the last 'x' days, where 'x' is the second number (argument).
3. The subdirectories of the tree of the given directory that were accessed during the last 'x' days, where 'x' is the second number (argument).
4. The files of the given directory for which all the users have read access.
5. The subdirectories of the given directory for which only the owner has complete rights (create, rename, delete, access files); all the others can access files only if they know the name of the file. Before printing each list (1 to 5) above, print an appropriate heading indicating (among other things), the number of files (or subdirectories) that will be printed. The script should execute iteratively (as many times as the user wishes - for different directories) and at the end (before the final exit) summarize the total number of files (or subdirectories) found in each case (from 1 to 5), for all directories searched.

```
#!/usr/bin/env bash

# Function to display heading
print_heading() {
    echo -e "\n$1"
    echo "======================================"
}

# Initialize counters for the summary
total_files_perm=0
total_files_modified=0
total_dirs_accessed=0
total_files_readable=0
total_dirs_owner_restricted=0

while true; do
    # Accept two arguments
    if [ "$#" -lt 2 ]; then
        echo "Usage: $0 <permissions_octal> <days_modified>"
        exit 1
    fi
```

```
perm_arg=$1
days_arg=$2

# Ask the user for the directory name
read -p "Enter the directory name (absolute path): " dir

# Ensure directory exists
if [ ! -d "$dir" ]; then
    echo "Directory does not exist. Please try again."
    continue
fi

# 1. Files with specific permissions
files_with_perm=$(find "$dir" -type f -perm "$perm_arg")
count_files_perm=$(echo "$files_with_perm" | wc -l)
total_files_perm=$((total_files_perm + count_files_perm))
print_heading "Files with permissions $perm_arg ($count_files_perm)"
echo "$files_with_perm"

# 2. Files modified in the last 'x' days
files_modified=$(find "$dir" -type f -mtime "-$days_arg")
count_files_modified=$(echo "$files_modified" | wc -l)
total_files_modified=$((total_files_modified + count_files_modified))
print_heading "Files modified in the last $days_arg days ($count_files_modified)"
echo "$files_modified"

# 3. Subdirectories accessed in the last 'x' days
dirs_accessed=$(find "$dir" -type d -atime "-$days_arg")
count_dirs_accessed=$(echo "$dirs_accessed" | wc -l)
total_dirs_accessed=$((total_dirs_accessed + count_dirs_accessed))
print_heading "Subdirectories accessed in the last $days_arg days ($count_dirs_accessed)"
echo "$dirs_accessed"

# 4. Files readable by all users
files_readable=$(ls -lR "$dir" | grep -E "^.r..r..r.." | awk '{print $9}')
count_files_readable=$(echo "$files_readable" | wc -l)
total_files_readable=$((total_files_readable + count_files_readable))
print_heading "Files readable by all users ($count_files_readable)"
echo "$files_readable"

# 5. Subdirectories with restricted owner permissions
dirs_owner_restricted=$(ls -ld "$dir"/*/ 2>/dev/null | grep -E "^drwx------" | awk '{print $9}')
count_dirs_owner_restricted=$(echo "$dirs_owner_restricted" | wc -l)
total_dirs_owner_restricted=$((total_dirs_owner_restricted + count_dirs_owner_restricted))
print_heading "Subdirectories restricted to owner only ($count_dirs_owner_restricted)"
echo "$dirs_owner_restricted"

# Ask if the user wants to continue
read -p "Do you want to search another directory? (y/n): " choice
if [[ "$choice" != "y" && "$choice" != "Y" ]]; then
```

```
    break
  fi
done

# Print summary
print_heading "Summary"
echo "Total files with permissions $perm_arg: $total_files_perm"
echo "Total files modified in the last $days_arg days: $total_files_modified"
echo "Total subdirectories accessed in the last $days_arg days: $total_dirs_accessed"
echo "Total files readable by all users: $total_files_readable"
echo "Total subdirectories restricted to owner only: $total_dirs_owner_restricted"

echo "Script finished."
```

Output

```
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./searching.sh 755 30
Enter the directory name (absolute path): /Users/danielistrati/Documents/UNIWA/Network Programming

Files with permissions 755 (       1)
====================================


Files modified in the last 30 days (      20)
====================================
/Users/danielistrati/Documents/UNIWA/Network Programming/.DS_Store
/Users/danielistrati/Documents/UNIWA/Network Programming/Istrati Daniel Network Report.pdf
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/.DS_Store
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/out/production/Test/cities.txt
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/out/production/Test/GUIClient.class
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/out/production/Test/GUIClient$1.class
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/out/production/Test/EntityClass.class
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/out/production/Test/Main.class
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/out/production/Test/namedEntities.txt
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/out/production/Test/CityClass.class
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/.idea/workspace.xml
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/src/cities.txt
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/src/GUIClient.java
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/src/.DS_Store
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/src/serverapp.java
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/src/Main.java
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/src/cities1.txt
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/src/namedEntities.txt
/Users/danielistrati/Documents/UNIWA/Network Programming/Istrati Daniel Network Report.docx
/Users/danielistrati/Documents/UNIWA/Network Programming/Screen Recording 2024-12-29 at 00.05.15.mov

Subdirectories accessed in the last 30 days (       4)
====================================
/Users/danielistrati/Documents/UNIWA/Network Programming
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/out/production/Test
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/.idea
/Users/danielistrati/Documents/UNIWA/Network Programming/Test/src

Files readable by all users (      29)
====================================
AppendixA_java_for_few_GlassFish_Installation.pdf
Assignment.pdf
Istrati
Istrati
Screen
```

```
Istrati
Istrati
Screen
Screenshot
Screenshot
Screenshot
Test
html_forms.pdf
http.pdf
Test.iml
out
src
production
Test
CityClass.class
EntityClass.class
GUIClient$1.class
GUIClient.class
Main.class
cities.txt
namedEntities.txt
GUIClient.java
Main.java
cities.txt
cities1.txt
namedEntities.txt
serverapp.java

Subdirectories restricted to owner only (       1)
==================================

Do you want to search another directory? (y/n): n

Summary
==================================
Total files with permissions 755: 1
Total files modified in the last 30 days: 20
Total subdirectories accessed in the last 30 days: 4
Total files readable by all users: 29
Total subdirectories restricted to owner only: 1
Script finished.
Daniels-MacBook-Pro:Lab_plus danielistrati$
```

B. Write a script called **teldb** that will manage a telephone catalogue that will be implemented in a file named **catalog**. The script should make the following:

- With the -a parameter it will add a new entry to the catalogue (i.e. to the catalog file). The addition will take place after the user asks for a name, surname, city, and the listing will be placed in a line (eg John Markou Peristeri 2105546789).
- With the -l parameter it will print the contents of the catalogue (with its numbered lines and omitting any blank lines).
- With the -s parameter followed by a number it will print the contents of the catalogue sorted by the corresponding column number (e.g. telcat -s 3 will display the contents sorted by city).
- With the -c parameter followed by a keyword it will only show us the directory lines that contain the keyword.
- With the -d parameter followed by a keyword and -b or -r, it will delete the directory line containing

the keyword. If the third parameter is -b it will insert in the position of each deleted line a blank line otherwise (-r) no.

- With the -n parameter it will print the number of blank lines in the catalogue, ask the user if he wants to delete these lines or not and act accordingly.

In any other case, an appropriate 'Usage' (indicating the correct usage of the script) message will be printed. Also, in the case of –c and –d parameters, if there are no lines containing the keyword the user should be notified with the appropriate message.

```
#!/bin/bash

CATALOG="catalog"

# Ensure the catalog file exists
if [ ! -f "$CATALOG" ]; then
    touch "$CATALOG"
fi

# Function to display usage instructions
usage() {
    echo "Usage: $0 -a | -l | -s <column> | -c <keyword> | -d <keyword> -b|-r | -n"
    echo "-a              Add a new entry to the catalog."
    echo "-l              Print the contents of the catalog with numbered lines."
    echo "-s <column>     Sort the catalog by the specified column (1: name, 2: surname, etc.)."
    echo "-c <keyword>    Show lines containing the keyword."
    echo "-d <keyword> -b|-r  Delete lines containing the keyword. Use -b to replace with blank lines, -r to remove entirely."
    echo "-n              Count blank lines and prompt to delete them."
    exit 1
}

# Add an entry to the catalog
add_entry() {
    read -p "Enter name: " name
    read -p "Enter surname: " surname
    read -p "Enter city: " city
    read -p "Enter phone number: " phone
    echo "$name $surname $city $phone" >> "$CATALOG"
    echo "Entry added successfully."
}

# Print the catalog with numbered lines
list_catalog() {
    nl -ba "$CATALOG" | sed '/^[[:space:]]*$/d'
}

# Sort the catalog by a specific column
sort_catalog() {
    column=$1
    sort -k "$column" "$CATALOG" | sed '/^[[:space:]]*$/d'
```

```
}

# Show lines containing a keyword
search_keyword() {
    keyword=$1
    grep -i "$keyword" "$CATALOG" || echo "No lines containing '$keyword' were found."
}

# Delete lines containing a keyword
delete_keyword() {
    keyword=$1
    mode=$2

    if ! grep -q "$keyword" "$CATALOG"; then
        echo "No lines containing '$keyword' were found."
        return
    fi

    if [ "$mode" = "-b" ]; then
        sed -i.bak "/$keyword/ s/.*/ /" "$CATALOG"
        echo "Lines containing '$keyword' replaced with blank lines."
    elif [ "$mode" = "-r" ]; then
        sed -i.bak "/$keyword/d" "$CATALOG"
        echo "Lines containing '$keyword' removed."
    else
        usage
    fi
}

# Count blank lines and optionally delete them
count_blank_lines() {
    count=$(grep -c '^$' "$CATALOG")
    echo "The catalog contains $count blank lines."
    if [ "$count" -gt 0 ]; then
        read -p "Do you want to delete them? (y/n): " choice
        if [[ "$choice" == "y" || "$choice" == "Y" ]]; then
            sed -i.bak '/^$/d' "$CATALOG"
            echo "Blank lines removed."
        else
            echo "No changes made."
        fi
    fi
}

# Main script logic
if [ "$#" -eq 0 ]; then
    usage
fi

case "$1" in
```

```
  -a)
    add_entry
    ;;
  -l)
    list_catalog
    ;;
  -s)
    if [ -n "$2" ]; then
      sort_catalog "$2"
    else
      usage
    fi
    ;;
  -c)
    if [ -n "$2" ]; then
      search_keyword "$2"
    else
      usage
    fi
    ;;
  -d)
    if [ -n "$2" ] && [ -n "$3" ]; then
      delete_keyword "$2" "$3"
    else
      usage
    fi
    ;;
  -n)
    count_blank_lines
    ;;
  *)
    usage
    ;;
esac
```

Output

```
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./teldb.sh
Usage: ./teldb.sh -a | -l | -s <column> | -c <keyword> | -d <keyword> -b|-r | -n
 -a              Add a new entry to the catalog.
 -l              Print the contents of the catalog with numbered lines.
 -s <column>     Sort the catalog by the specified column (1: name, 2: surname, etc.).
 -c <keyword>    Show lines containing the keyword.
 -d <keyword> -b|-r  Delete lines containing the keyword. Use -b to replace with blank lines, -r to remove entirely.
 -n              Count blank lines and prompt to delete them.
```

```
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./teldb.sh -a
Enter name: John
Enter surname: Doe
Enter city: Cahul
Enter phone number: +11234123
Entry added successfully.
```

```
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./teldb.sh -l
     1  Dan Balan Chisinau +373777777
     2  Ivan Tolika Balti +373000000
     3  Ion Popescu Chisinau +373123123
     4  Maria Sandu Balti +373123456
     5  Vasile Ionescu Cahul +373987654
     6  Ana Ciobanu Orhei +373456456
     7  Mihai Rusu Soroca +373789789John Doe Cahul +11234123
     8  John Doe Cahul +11234123
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./teldb.sh -s 1
Ana Ciobanu Orhei +373456456
Dan Balan Chisinau +373777777
Ion Popescu Chisinau +373123123
Ivan Tolika Balti +373000000
John Doe Cahul +11234123
Maria Sandu Balti +373123456
Mihai Rusu Soroca +373789789John Doe Cahul +11234123
Vasile Ionescu Cahul +373987654
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./teldb.sh -c Chisinau
 Dan Balan Chisinau +373777777
 Ion Popescu Chisinau +373123123
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./teldb.sh -d Balti -r
Lines containing 'Balti' removed.
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./teldb.sh -l
     1  Dan Balan Chisinau +373777777
     2  Ion Popescu Chisinau +373123123
     3  Vasile Ionescu Cahul +373987654
     4  Ana Ciobanu Orhei +373456456
     5  Mihai Rusu Soroca +373789789
     6  John Doe Cahul +11234123
Daniels-MacBook-Pro:Lab_plus danielistrati$ ./teldb.sh -n
The catalog contains 0 blank lines.
```

C. Write a script called cmpdir that compares the contents of two directories (whose names are given as arguments - and first the script should check if they are actually directories) with regard to the files they contain. As a result it will initially print for each directory separately, how many and which files are not in the other directory, and what their total size is. It will then print how many and what are the two directories' common files and their total size. Finally, it will move all common files of the two directories to a third directory (which will also be given / checked as an argument), and will create appropriate *hard links* from the two directories to them.

```bash
#!/bin/bash

# Check if a path is a valid directory
check_directory() {
    if [ ! -d "$1" ]; then
        echo "Error: $1 is not a valid directory."
        exit 1
    fi
}

# Calculate the total size of files
calculate_size() {
```

```
    local files=("$@")

    # If there are no files, return 0
    if [ "${#files[@]}" -eq 0 ]; then
        echo 0
        return
    fi

    # Use `du` safely with filenames containing spaces
    total_size=0
    for file in "${files[@]}"; do
        if [ -f "$file" ]; then
            file_size=$(du -b "$file" 2>/dev/null | awk '{print $1}')
            total_size=$((total_size + file_size))
        fi
    done

    echo "$total_size"
}


# Ensure correct usage
if [ "$#" -ne 3 ]; then
    echo "Usage: $0 <directory1> <directory2> <destination_directory>"
    exit 1
fi

# Assign arguments to variables
dir1=$1
dir2=$2
dest_dir=$3

# Check if input directories exist
check_directory "$dir1"
check_directory "$dir2"

# Check or create the destination directory
if [ ! -d "$dest_dir" ]; then
    mkdir -p "$dest_dir" || { echo "Error: Could not create destination directory $dest_dir."; exit 1; }
fi

files_dir1=()
while IFS= read -r file; do
    files_dir1+=("$file")
done < <(find "$dir1" -maxdepth 1 -type f -exec basename {} \;)

files_dir2=()
while IFS= read -r file; do
    files_dir2+=("$file")
done < <(find "$dir2" -maxdepth 1 -type f -exec basename {} \;)
```

```
# Initialize arrays
unique_dir1=()
unique_dir2=()
common_files=()

# Compare files between the two directories
for file in "${files_dir1[@]}"; do
    if [[ " ${files_dir2[@]} " =~ " $file " ]]; then
        # Check if contents are identical
        hash1=$(shasum "$dir1/$file" | awk '{print $1}')
        hash2=$(shasum "$dir2/$file" | awk '{print $1}')
        if [[ "$hash1" == "$hash2" ]]; then
            common_files+=("$file")
        else
            unique_dir1+=("$dir1/$file")  # Same filename, different content
            unique_dir2+=("$dir2/$file")  # Same filename, different content
        fi
    else
        unique_dir1+=("$dir1/$file")
    fi
done

# Check for unique files in dir2
for file in "${files_dir2[@]}"; do
    if [[ ! " ${files_dir1[@]} " =~ " $file " ]]; then
        unique_dir2+=("$dir2/$file")
    fi
done

# Print unique files in dir1
echo "Files unique to $dir1 (${#unique_dir1[@]} files):"
printf "%s\n" "${unique_dir1[@]}"
size_dir1=$(calculate_size "${unique_dir1[@]}")
echo "Total size: $size_dir1 bytes"

# Print unique files in dir2
echo "Files unique to $dir2 (${#unique_dir2[@]} files):"
printf "%s\n" "${unique_dir2[@]}"
size_dir2=$(calculate_size "${unique_dir2[@]}")
echo "Total size: $size_dir2 bytes"

# Print common files
echo "Common files between $dir1 and $dir2 (${#common_files[@]} files):"
printf "%s\n" "${common_files[@]}"
common_files_paths=()
for file in "${common_files[@]}"; do
    common_files_paths+=("$dir1/$file")
done
```

```
size_common=$(calculate_size "${common_files_paths[@]}")
echo "Total size: $size_common bytes"

# Move common files to the destination directory and create hard links
echo "Moving common files to $dest_dir and creating hard links..."
for file in "${common_files[@]}"; do
    mv "$dir1/$file" "$dest_dir/"
    ln "$dest_dir/$file" "$dir1/"
    ln "$dest_dir/$file" "$dir2/"
done
echo "Operation complete."

exit 0
```

```
bash-5.2$ ./cmpdir.sh "/Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/comp1" "/Users/danielistrati/Documents/UNIWA/Operating Systems
/Lab_plus/comp2" "/Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/compcommon"
Files unique to /Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/comp1 (1 files):
/Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/comp1/what?.txt
Total size: 0 bytes
Files unique to /Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/comp2 (1 files):
/Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/comp2/Different.txt
Total size: 0 bytes
Common files between /Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/comp1 and /Users/danielistrati/Documents/UNIWA/Operating Systems
/Lab_plus/comp2 (2 files):
Again.txt
Hello.txt
Total size: 0 bytes
Moving common files to /Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/compcommon and creating hard links...
ln: /Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/comp2//Again.txt: File exists
ln: /Users/danielistrati/Documents/UNIWA/Operating Systems/Lab_plus/comp2//Hello.txt: File exists
Operation complete.
```

D. Write a script called bck that will get for a specific user (whose username is given as the *first argument*) a *backup copy* of one area of his account to another. The script should accept a directory (or file) as the *second argument*, create a *backup copy* of the area specified by this argument (with use of *tar* command), and copy it to the directory given as the *third argument*. If however the third argument is a file (and not a directory) then it should simply append the *backup copy* to that file. Next, modify script bck properly (naming it bck1) to perform the requested backup scheduled (with use of at command) at a specific *time* of your choice (try to give the *time* as an argument as well).

```
#!/bin/bash

# Ensure correct number of arguments
if [ "$#" -ne 3 ]; then
    echo "Usage: $0 <username> <source_path> <destination_path>"
    exit 1
fi

# Assign arguments to variables
username="$1"
source_path="$2"
destination="$3"

# Check if the user exists
if ! id "$username" &>/dev/null; then
```

```
    echo "Error: User '$username' does not exist."
    exit 1
fi

# Check if the source exists
if [ ! -e "$source_path" ]; then
    echo "Error: Source '$source_path' does not exist."
    exit 1
fi

# Generate a timestamped backup filename
timestamp=$(date +"%Y%m%d_%H%M%S")
backup_file="/tmp/${username}_backup_${timestamp}.tar.gz"

# Create the tar archive
tar -czf "$backup_file" "$source_path"
echo "Backup created: $backup_file"

# Check if the destination is a directory
if [ -d "$destination" ]; then
    cp "$backup_file" "$destination/"
    echo "Backup copied to directory: $destination"
elif [ -f "$destination" ]; then
    cat "$backup_file" >> "$destination"
    echo "Backup appended to file: $destination"
else
    echo "Error: Destination '$destination' is neither a directory nor a file."
    exit 1
fi

exit 0
```

```
bash-5.2$ ./bck.sh danielistrati /Users/danielistrati/Documents/photos /Users/danielistrati/Documents/Backups
tar: Removing leading '/' from member names
Backup created: /tmp/danielistrati_backup_20250130_225351.tar.gz
Backup copied to directory: /Users/danielistrati/Documents/Backups
```