Sistemas Operativos **Ano lectivo 2025/2026**

Trabalho Prático 1

Linux Recycle Bin Simulation – Support Resources[1]

---

1. Sample Metadata Format

metadata.db Structure (CSV Format)

```
ID,ORIGINAL_NAME,ORIGINAL_PATH,DELETION_DATE,FILE_SIZE,FILE_TYPE,PERMIS-
SIONS,OWNER
1696234567_abc123,document.txt,/home/user/Documents/document.txt,2024-10-
02 14:30:22,4096,file,644,user:user
1696234890_def456,project_folder,/home/user/Projects/project_folder,2024-
10-02 15:45:10,20480,directory,755,user:user
```

Field Descriptions:

- **ID**: Unique identifier (timestamp_randomstring)
- **ORIGINAL_NAME**: Original filename/directory name
- **ORIGINAL_PATH**: Complete absolute path where file was located
- **DELETION_DATE**: When file was moved to recycle bin (YYYY-MM-DD HH:MM:SS)
- **FILE_SIZE**: Size in bytes
- **FILE_TYPE**: "file" or "directory"
- **PERMISSIONS**: Original permission bits (e.g., 644, 755)
- **OWNER**: Original owner and group (user:group format)

---

2. Basic Script Template

Starter Code Structure

```
#!/bin/bash
```

[1] The text of this project proposal had AI contributions to its completion.

```
#################################################
# Linux Recycle Bin Simulation
# Author: [Your Name]
# Date: [Date]
# Description: Shell-based recycle bin system
#################################################

# Global Configuration
RECYCLE_BIN_DIR="$HOME/.recycle_bin"
FILES_DIR="$RECYCLE_BIN_DIR/files"
METADATA_FILE="$RECYCLE_BIN_DIR/metadata.db"
CONFIG_FILE="$RECYCLE_BIN_DIR/config"

# Color codes for output (optional)
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

################################################
# Function: initialize_recyclebin
# Description: Creates recycle bin directory structure
# Parameters: None
# Returns: 0 on success, 1 on failure
################################################
initialize_recyclebin() {
    if [ ! -d "$RECYCLE_BIN_DIR" ]; then
        mkdir -p "$FILES_DIR"
        touch "$METADATA_FILE"
        echo "# Recycle Bin Metadata" > "$METADATA_FILE"
        echo "ID,ORIGINAL_NAME,ORIGINAL_PATH,DELE-
TION_DATE,FILE_SIZE,FILE_TYPE,PERMISSIONS,OWNER" >> "$METADATA_FILE"
        echo "Recycle bin initialized at $RECYCLE_BIN_DIR"
        return 0
    fi
    return 0
}

################################################
# Function: generate_unique_id
# Description: Generates unique ID for deleted files
# Parameters: None
# Returns: Prints unique ID to stdout
################################################
generate_unique_id() {
    local timestamp=$(date +%s)
    local random=$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 6 | head -
n 1)
    echo "${timestamp}_${random}"
}

################################################
# Function: delete_file
# Description: Moves file/directory to recycle bin
# Parameters: $1 - path to file/directory
# Returns: 0 on success, 1 on failure
################################################
delete_file() {
```

```
    # TODO: Implement this function
    local file_path="$1"

    # Validate input
    if [ -z "$file_path" ]; then
        echo -e "${RED}Error: No file specified${NC}"
        return 1
    fi

    # Check if file exists
    if [ ! -e "$file_path" ]; then
        echo -e "${RED}Error: File '$file_path' does not exist${NC}"
        return 1
    fi

    # Your code here
    # Hint: Get file metadata using stat command
    # Hint: Generate unique ID
    # Hint: Move file to FILES_DIR with unique ID
    # Hint: Add entry to metadata file

    echo "Delete function called with: $file_path"
    return 0
}

################################################
# Function: list_recycled
# Description: Lists all items in recycle bin
# Parameters: None
# Returns: 0 on success
################################################
list_recycled() {
    # TODO: Implement this function
    echo "=== Recycle Bin Contents ==="

    # Your code here
    # Hint: Read metadata file and format output
    # Hint: Use printf for formatted table
    # Hint: Skip header line

    return 0
}

################################################
# Function: restore_file
# Description: Restores file from recycle bin
# Parameters: $1 - unique ID of file to restore
# Returns: 0 on success, 1 on failure
################################################
restore_file() {
    # TODO: Implement this function
    local file_id="$1"

    if [ -z "$file_id" ]; then
        echo -e "${RED}Error: No file ID specified${NC}"
        return 1
    fi

    # Your code here
```

```
    # Hint: Search metadata for matching ID
    # Hint: Get original path from metadata
    # Hint: Check if original path exists
    # Hint: Move file back and restore permissions
    # Hint: Remove entry from metadata

    return 0
}

################################################
# Function: empty_recyclebin
# Description: Permanently deletes all items
# Parameters: None
# Returns: 0 on success
################################################
empty_recyclebin() {
    # TODO: Implement this function

    # Your code here
    # Hint: Ask for confirmation
    # Hint: Delete all files in FILES_DIR
    # Hint: Reset metadata file

    return 0
}

################################################
# Function: search_recycled
# Description: Searches for files in recycle bin
# Parameters: $1 - search pattern
# Returns: 0 on success
################################################
search_recycled() {
    # TODO: Implement this function
    local pattern="$1"

    # Your code here
    # Hint: Use grep to search metadata

    return 0
}

################################################
# Function: display_help
# Description: Shows usage information
# Parameters: None
# Returns: 0
################################################
display_help() {
    cat << EOF
Linux Recycle Bin - Usage Guide

SYNOPSIS:
    $0 [OPTION] [ARGUMENTS]

OPTIONS:
    delete <file>       Move file/directory to recycle bin
    list                List all items in recycle bin
    restore <id>        Restore file by ID
```

```
    search <pattern>    Search for files by name
    empty               Empty recycle bin permanently
    help                Display this help message

EXAMPLES:
    $0 delete myfile.txt
    $0 list
    $0 restore 1696234567_abc123
    $0 search "*.pdf"
    $0 empty

EOF
    return 0
}

################################################
# Function: main
# Description: Main program logic
# Parameters: Command line arguments
# Returns: Exit code
################################################
main() {
    # Initialize recycle bin
    initialize_recyclebin

    # Parse command line arguments
    case "$1" in
        delete)
            shift
            delete_file "$@"
            ;;
        list)
            list_recycled
            ;;
        restore)
            restore_file "$2"
            ;;
        search)
            search_recycled "$2"
            ;;
        empty)
            empty_recyclebin
            ;;
        help|--help|-h)
            display_help
            ;;
        *)
            echo "Invalid option. Use 'help' for usage information."
            exit 1
            ;;
    esac
}

# Execute main function with all arguments
main "$@"
```

3. Useful Shell Commands Reference

File Operations

```
# Get file information
stat -c "%n %s %a %U:%G" filename          # Name, size, permissions, ow-
ner
file filename                               # Determine file type
du -sb directory                            # Get directory size

# Move/Copy operations
mv source destination                       # Move file
cp -a source destination                    # Copy with attributes
cp -r source destination                    # Recursive copy

# Safe removal
rm -rf directory                            # Remove directory recursively
```

String and Date Operations

```
# Date formatting
date +%s                                    # Unix timestamp
date "+%Y-%m-%d %H:%M:%S"                  # Formatted date
date -d "2024-10-01" +%s                   # Convert date to timestamp

# String manipulation
basename /path/to/file.txt                  # Returns: file.txt
dirname /path/to/file.txt                   # Returns: /path/to
realpath file.txt                           # Get absolute path

# Generate random strings
cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n 1
```

File Reading and Parsing

```
# Read CSV file line by line
while IFS=',' read -r col1 col2 col3; do
    echo "Column 1: $col1"
done < file.csv

# Skip header line
tail -n +2 file.csv

# Search in file
grep "pattern" file.txt
grep -v "^#" file.txt                       # Exclude comments
```

Conditional Checks

```
# File existence checks
[ -e file ]      # File exists
[ -f file ]      # Is regular file
[ -d dir ]       # Is directory
[ -r file ]      # Is readable
[ -w file ]      # Is writable

# String checks
[ -z "$var" ]    # String is empty
[ -n "$var" ]    # String is not empty
[ "$a" = "$b" ]  # Strings are equal
```

4. Common Pitfalls and Solutions

Issue 1: Handling Spaces in Filenames

Problem: File paths with spaces break the script

Solution:

```
# Always quote variables
delete_file "$file_path"

# Use arrays for multiple files
files=("file1.txt" "file with spaces.txt")
for file in "${files[@]}"; do
    echo "$file"
done
```

Issue 2: Preserving File Metadata

Problem: Files lose permissions when moved

Solution:

```
# Get original permissions
original_perms=$(stat -c "%a" "$file")

# Store in metadata
echo "$id,$filename,$path,$date,$size,$type,$original_perms,$owner" >>
"$METADATA_FILE"

# Restore permissions
chmod "$original_perms" "$restored_file"
```

Issue 3: Handling Duplicate Filenames

Problem: Multiple files with same name cause conflicts

Solution:

```
# Use unique ID for storage
unique_id=$(generate_unique_id)
mv "$original_file" "$FILES_DIR/$unique_id"

# Store original name in metadata for restoration
```

Issue 4: Corrupted Metadata File

Problem: Script breaks if metadata is malformed

Solution:

```
# Validate metadata file exists and is readable
```

```
if [ ! -f "$METADATA_FILE" ] || [ ! -r "$METADATA_FILE" ]; then
    echo "Error: Metadata file is missing or unreadable"
    initialize_recyclebin
fi

# Backup metadata before modifications
cp "$METADATA_FILE" "$METADATA_FILE.bak"
```

---

5. Testing Checklist

Basic Functionality Tests

- [ ] Initialize recycle bin structure
- [ ] Delete a single file
- [ ] Delete multiple files
- [ ] Delete a directory with contents
- [ ] List empty recycle bin
- [ ] List recycle bin with items
- [ ] Restore a file to original location
- [ ] Empty recycle bin

Edge Case Tests

- [ ] Delete non-existent file
- [ ] Delete file without permissions
- [ ] Restore when original path is occupied
- [ ] Restore non-existent ID
- [ ] Handle filenames with spaces
- [ ] Handle filenames with special characters
- [ ] Delete very large files
- [ ] Delete symbolic links

Error Handling Tests

- [ ] Invalid command line arguments
- [ ] Insufficient disk space
- [ ] Corrupted metadata file
- [ ] Permission denied scenarios
- [ ] Empty recycle bin operations

---

6. Sample Test Script

```
#!/bin/bash
# test_recyclebin.sh - Automated testing script

echo "=== Testing Recycle Bin System ==="

# Test 1: Initialization
```

```
echo "Test 1: Initialization"
./recycle_bin.sh help
echo ""

# Test 2: Create test files
echo "Test 2: Creating test files"
mkdir -p test_data
echo "Sample content" > test_data/file1.txt
echo "Another file" > test_data/file2.txt
mkdir test_data/subdir
echo "Nested file" > test_data/subdir/file3.txt

# Test 3: Delete files
echo "Test 3: Deleting files"
./recycle_bin.sh delete test_data/file1.txt
./recycle_bin.sh delete test_data/subdir

# Test 4: List contents
echo "Test 4: Listing recycle bin"
./recycle_bin.sh list

# Test 5: Search
echo "Test 5: Searching for files"
./recycle_bin.sh search "file1"

# Cleanup
rm -rf test_data
echo ""
echo "=== Tests Complete ==="
```

---

7. Debugging Tips

Enable Debug Mode

```
# Run script with debugging
bash -x recycle_bin.sh delete file.txt

# Add debug output in script
set -x  # Enable debug mode
# ... your code ...
set +x  # Disable debug mode
```

Add Logging

```
LOG_FILE="$RECYCLE_BIN_DIR/recyclebin.log"

log_message() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" >> "$LOG_FILE"
}

# Usage
log_message "Deleted file: $filename"
```

Validate Script Syntax

```
# Check for syntax errors
bash -n recycle_bin.sh

# Use shellcheck for best practices
```

```
shellcheck recycle_bin.sh
```

---

8. Example Implementation Snippets

Reading Metadata Line by Line

```
list_recycled() {
    printf "%-20s %-40s %-20s %-10s\n" "ID" "ORIGINAL NAME" "DELETION
DATE" "SIZE"
    printf "%s\n" "-------------------------------------------------------
------------------------"

    tail -n +2 "$METADATA_FILE" | while IFS=',' read -r id name path date
size type perms owner; do
        printf "%-20s %-40s %-20s %-10s\n" "$id" "$name" "$date" "$size"
    done
}
```

Confirmation Prompt

```
confirm_action() {
    read -p "Are you sure? (y/n): " choice
    case "$choice" in
        y|Y|yes|YES) return 0 ;;
        *) return 1 ;;
    esac
}

# Usage
if confirm_action; then
    # Proceed with operation
    rm -rf "$FILES_DIR"/*
fi
```

Finding Metadata Entry

```
find_metadata_entry() {
    local search_id="$1"
    grep "^$search_id," "$METADATA_FILE"
}

# Usage
entry=$(find_metadata_entry "$file_id")
if [ -n "$entry" ]; then
    # Process entry
    echo "Found: $entry"
fi
```

---

9. Additional Resources

Online Documentation

- **Bash Reference Manual:** https://www.gnu.org/software/bash/manual/
- **Advanced Bash Scripting Guide:** https://tldp.org/LDP/abs/html/
- **ShellCheck Wiki:** https://www.shellcheck.net/wiki/

Recommended Reading

- "Classic Shell Scripting" by Arnold Robbins
- "Linux Command Line and Shell Scripting Bible"
- Man pages: `man bash`, `man stat`, `man mv`

Video Tutorials

- Search for: "Bash scripting tutorial"
- Search for: "Linux file operations"
- Search for: "Shell script functions"

---

10. Office Hours Schedule

When: Pedro Azevedo Fernandes: 10am-11am: 4.1.01; 15pm-16pm: 4.2.25.
What to Bring:

- Your current code
- Specific error messages
- Description of what you've tried

Discussion Forum:
https://elearning.ua.pt/mod/forum/view.php?id=1634485

---

11. Frequently Asked Questions

Q: Can I use other scripting languages?
A: No, this project must be implemented in Bash shell scripting.

Q: Can I use external libraries?
A: You should use standard Linux utilities only (mv, rm, grep, etc.).

Q: What if the original path no longer exists during restoration?
A: Your script should detect this and either create the directory structure or ask the user where to restore.

Q: How do I handle very large files?
A: Use efficient file operations. Consider checking available disk space before moving files.

Q: Can I modify the metadata format?
A: Yes, but document your changes clearly in your technical documentation.

Note: This document will be updated periodically. Check the course website for the latest version.
