readme

1. Project Titile and Overview

This project replaces the default `rm` command with a safer alternative that moves files to a trash directory (`/tmp/trash/`) instead of permanently deleting them. It also includes a `restore` script to recover deleted files, and a logging mechanism to track all deletions.

2. Features

- ✅ Safe deletion using a custom `rm` script
- 🔁 Restore files using the `restore` command
- 📂 Supports recursive deletion (`-r`) for directories
- 🧾 All deletions are logged in a CSV-formatted log file
- 🚫 Prevents accidental permanent deletions

3. Directory Structure

```
~/bin/
├── rm # Custom deletion script
├── restore # File recovery script
/tmp/trash/
├── trash.log # Deletion logs
├── ... # Moved (trashed) files
```

4. How it works

`rm` Script

- Moves files/directories to `/tmp/trash/`
- Appends a timestamp to filenames to avoid name conflicts
- Logs the following to `/tmp/trash/trash.log` in CSV format:
  - Original path
  - Trash path
  - Timestamp of deletion

`restore` Script

- Accepts a filename (or partial match)
- Finds the **most recent match** in `trash.log`
- Moves it back to its original location

- Creates destination folder if missing
- Warns if a conflict occurs during restoration

5. Usage Instructions

```
mkdir -p ~/bin
mv rm restore ~/bin/
chmod +x ~/bin/rm ~/bin/restore
echo 'export PATH="$HOME/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

Delete files:

```
rm file.txt
rm -r my_folder/
```

Restore files:

```
restore file.txt
```

Check trash:

```
ls /tmp/trash/
cat /tmp/trash/trash.log
```

6. Edge Cases Handled

- ❌ Prevents deleting directories without `-r`
- 🧪 Verifies if a file or directory exists before "deletion"
- 🛑 Rejects unsupported flags (like `-f`, `-z`, etc.)
- ⚠️ Checks for restore conflicts (renames if destination already exists)

7. Example

```
# Create and delete a file
echo "hello" > hello.txt
rm hello.txt

# Delete directory
mkdir testdir && touch testdir/test.txt
rm -r testdir
```

```
# Restore file
restore hello.txt
```

8. Demostration

See attached video showing:

- Deletion of file and folder
- Trash contents
- Log file entries
- Restoration of a file

rm

```
#!/bin/bash

TRASH_DIR="/tmp/trash"
LOG_FILE="$TRASH_DIR/trash.log"
TIMESTAMP=$(date +%Y%m%dT%H%M%S)

# Check if trash directory exists
[ -d "$TRASH_DIR" ] || mkdir -p "$TRASH_DIR"
touch "$LOG_FILE"

# Usage message
usage() {
  echo "Usage: rm [-r] file1 [file2 ...]"
  exit 1
}

# Parse options
recursive=false
files=()

while [[ "$#" -gt 0 ]]; do
  case "$1" in
    -r) recursive=true ;;
    -*) echo "Unsupported option: $1"; usage ;;
    *) files+=("$1") ;;
  esac
  shift
done

# Ensure at least one file
[ "${#files[@]}" -eq 0 ] && usage

for target in "${files[@]}"; do
  if [ ! -e "$target" ]; then
```

```
    echo "Error: $target does not exist"
    continue
  fi

  if [ -d "$target" ] && [ "$recursive" ≠ true ]; then
    echo "Error: '$target' is a directory. Use -r to delete."
    continue
  fi

  filename=$(basename "$target")
  trash_path="$TRASH_DIR/${filename}_$TIMESTAMP"

  mv "$target" "$trash_path"
  echo "$target,$trash_path,$(date '+%Y-%m-%d %H:%M:%S')" >> "$LOG_FILE"
done
```

restore

```
#!/bin/bash

TRASH_DIR="/tmp/trash"
LOG_FILE="$TRASH_DIR/trash.log"

[ -f "$LOG_FILE" ] || { echo "Trash log not found."; exit 1; }

query="$1"
[ -z "$query" ] && { echo "Usage: restore filename"; exit 1; }

# Find most recent match in log
entry=$(grep "$query" "$LOG_FILE" | tail -n 1)

if [ -z "$entry" ]; then
  echo "No matching file found in trash."
  exit 1
fi

IFS=',' read -r original_path trash_path timestamp <<< "$entry"

# Handle destination path
dest_dir=$(dirname "$original_path")
[ -d "$dest_dir" ] || mkdir -p "$dest_dir"

# Restore, check for conflicts
if [ -e "$original_path" ]; then
  echo "Conflict: File already exists at $original_path"
  new_path="${original_path}_restored_$(date +%s)"
  mv "$trash_path" "$new_path"
  echo "Restored as $new_path"
```

```
else
  mv "$trash_path" "$original_path"
  echo "Restored to $original_path"
fi
```