FILER

Introduction

In Linux, Bash is a powerful tool with many commands, but it doesn't always make file management easy. That's where Filer comes in, a simple Bash-based tool designed to manage files while backing up, or when in need of organizing.

Objective

The goal of this project is to simplify common file management tasks, such as backing up and organizing files. These tasks often require complex commands or manual effort, which Filer streamlines with easy-to-use sub-commands, making file handling faster and more efficient for Linux users.

Methodology

Filer is designed to run under linux which is why it's written in bash, a language that is popularly used in linux to write scripts that are efficient and easy to read.

List of tools used to create Filer:

1. Bash
2. GNU coreutils - awk, ls, chmod, etc..
3. Linux Operating system
4. Git - For version control

Filer provides two important features

- backup/restore
- organize

Backup/Restore - Allows the user to backup and restore files easily. To use this feature, simply run:

$ filer backup hello.txt

Similarly, run the restore command to restore the file back to its original state

$ filer restore hello.txt

Organize - This is the most prominent feature of Filer which allows the user to organize given files by putting them in their respective directories based on their file extension. To use this feature, simply run

$ filer organize hello.txt app.apk chat.txt presentation.ppt

Filer will create directories such as txt, apk and ppt, and then put the files accordingly.

Code and Implementation Details

Filer's source code is written purely in bash and is managed by a widely used version control system known as git. The source code contains several bash scripts, mainly, an installer script, uninstaller script and the filer command itself. There are also test scripts to perform unit tests.

Implementation features

1. Proper bash guidelines and idioms

The scripts used in Filer's source code are written in pure bash and follow proper bash guidelines that are easy to read.

- It uses conditionals such as if and uses loops such as for loops.
- Uses functions for reducing code duplication
- Input/Output handling. Ex: 2>/dev/null
- Uses command substitution

2. Use of shebangs

Filer makes use of shebangs to simplify running scripts as commands under linux environment.

#!/usr/bin/bash - is used quite commonly. Which provide several benefits such as

- Can be executed directly by running the file. Ex ./filer
- No need for .sh extension

3. GNU Coreutils

The GNU Core Utilities are the basic file, shell and text manipulation utilities of the GNU operating system. These are the core utilities which are expected to exist on every operating system.

Filer makes use of some of these utils such as

- ls - for listing files
- awk - for text manipulation
- mktemp - for creation of temporary directories
- bash - as a scripting language
- chmod - for permission handling
- rm - for removing files
- cp - for copying files
- mkdir - for creation of directories
- mv - for moving files from one directory to another

4. Unit tests

Filer performs tests on all of its features to make sure they work as intended.

These tests are also written in bash which run under a temporary directory in an isolated fashion.

Tests can be run by executing ./tests.sh from the project root.

Installation/Uninstallation

Filer can be installed by simply running ./install.sh on a linux operating system.

1. Clone the git repository
2. Cd to the project root
3. Run ./install.sh

To uninstall filer, follow first 2 steps and then run ./uninstall.sh

Tests

Filer can be tested by running ./tests.sh which will perform tests on the command by running it in a temporary directory in isolation. This also prevents interference with the environment during the testing phase.

Code Example

A snippet from Filer's source code which is used to iterate through all the files and create a backup

```
74  # Backup the given file(s) by iterating through them using
75  # a for loop
76  for f in $@; do
77
78    # Check if given file exists before backing up
79    if [ -e "$f" ]; then
80      mv "$f" "$f.bak" || exit 1
81      echo "Backed up $f"
82    else
83      echo "Error: no such file or directory $f"
84    fi
85  done
86
```

Here, **$@** represents a list of all the arguments passed to Filer, which in this context refers to a list of files that are to be backed up.

if [-e "$f"] is used to check if the file $f exists or not. If it exists, then rename it using the move command.

|| exit 1 is used to exit if any failure occurs while renaming the file that Filer cannot handle

Use Cases

Scenario 1:

Filer has several use cases among general audience and linux enthusiasts. Let's for example, take a college student who has several documents all clustered over in the Downloads folder. The documents include, powerpoints (.ppt), word documents (.doc/docx), text files (.txt), source code (.py), music (.mp3) etc..

Of course, one can manually arrange all the files themselves but filer's organize feature does that in one go.

Just run **$ filer organize file1 file2 file3…** and all the files will be organized accordingly.

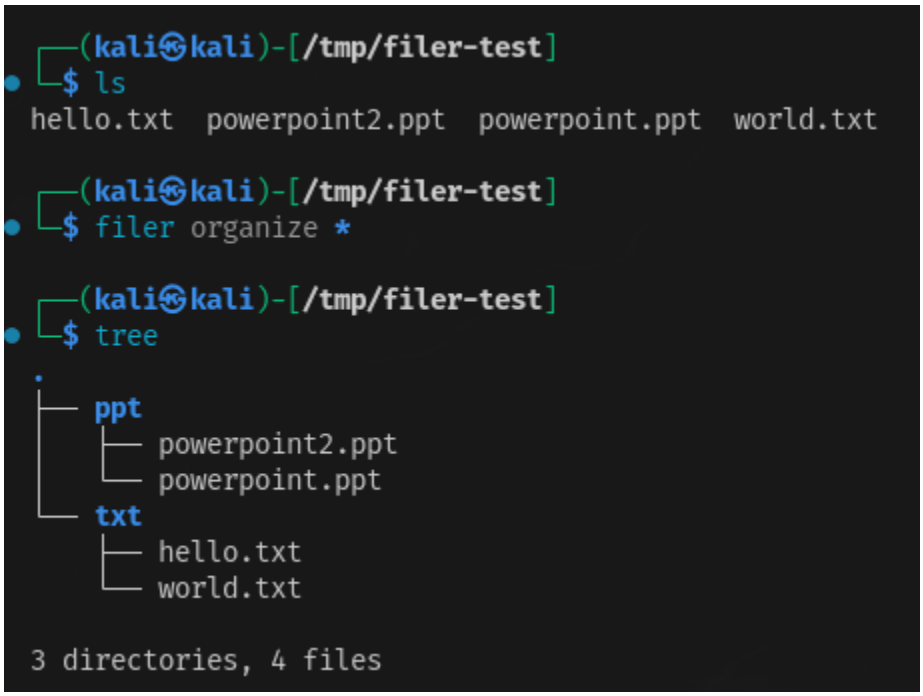
Scenario 2:

A user may want to edit a config file located in ~/.config/abc.cfg but doing so will destroy the previous configuration. Here, the user can simply run

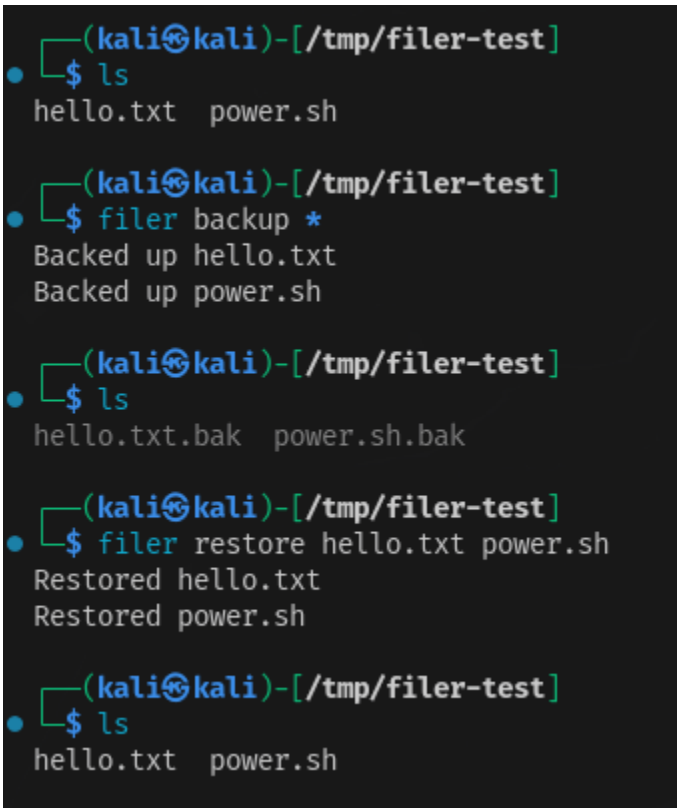
$ filer backup ~/.config/abc.cfg which will automatically create a backup of that file.

Later, the file can be restored by running **$ filer backup ~/.config/abc.cfg**

Usage Demo (organize)



Usage Demo (backup/restore)



Results and Observations

Filer command has been tested on Kali Linux. It passes all the tests with good performance.

Consider test **test_organize_speed** which tests filer's speed when it comes to organizing thousands of files. Filer can organize more than 1000 files in approximately 5s

```
real    0m5.998s
user    0m4.702s
sys     0m1.908s
++ return 0
+ output='Organizing  1378 Files'
+ exitcode=0
+ cd /home/kali/filer
+ echo '######################'
######################
+ echo 'Ran test test_organize_speed'
Ran test test_organize_speed
+ '[' 0 -eq 0 ']'
+ echo 'Success ✅'
Success ✅
+ echo 'Output: Organizing  1378 Files'
Output: Organizing  1378 Files
+ rm -r /tmp/tmp.HlQshjOMOP
+ return 0
+ count=4
+ echo 'Total succedeed tests: 4'
Total succedeed tests: 4
```

Similarly on testing **test_backup_speed,** filer can backup more than 1000 files in under ~1.5s

```
real    0m1.531s
user    0m1.096s
sys     0m0.495s
++ return 0
+ output='Backup  1378 Files'
+ exitcode=0
+ cd /home/kali/filer
+ echo '######################'
######################
+ echo 'Ran test test_backup_speed'
Ran test test_backup_speed
+ '[' 0 -eq 0 ']'
+ echo 'Success ✅'
Success ✅
+ echo 'Output: Backup  1378 Files'
Output: Backup  1378 Files
+ rm -r /tmp/tmp.Wrz5fZ089C
+ return 0
+ count=5
+ echo 'Total succedeed tests: 5'
Total succedeed tests: 5
```

Future Enhancements

As with every tool, filer also has several things it can improve on. Some of them are

- Custom folder names for organization
- A preview option to show file organization before moving
- Compress for backups
- Support for other unix-like systems or even windows

Conclusion

In conclusion, Filer stands as a practical and user-friendly Bash tool that addresses key gaps in Linux file management by simplifying backups, restores, and organization tasks. Through its straightforward sub-commands and reliance on standard GNU tools, it delivers efficient results, as shown in tests where it handles thousands of files in seconds with no issues. Developed as part of the SmartEd internship, this project not only makes daily file handling easier for students, developers, and admins but also demonstrates how simple scripting can solve real-world problems. With room for enhancements like custom folders and compression, Filer has strong potential to evolve into an even more versatile utility for Linux users.