Tutorial: Moving Files with Permissions Retained Using `moveperm.sh`

Transferring files between drives or directories often results in the loss of important file attributes such as permissions and ownership. This can lead to issues when the files are accessed later, especially when transferring data for backup purposes. In this tutorial, we will walk through a script that automates the process of moving files from one drive to another, ensuring that file permissions and ownership are preserved.

This script is particularly useful for backing up large amounts of data to external drives or between different partitions, while maintaining the correct permissions.

Why Use Rsync to Move Files with Permissions?

`rsync` is an efficient tool for transferring files while preserving important attributes like ownership and permissions. It's highly customizable and handles large transfers reliably, even in cases where the transfer is interrupted. Additionally, using `rsync` ensures that files are only transferred if they are new or modified, saving time and bandwidth.

By using `rsync`, we can: - **Maintain File Ownership**: Ensuring the correct user has ownership of the files. - **Preserve Permissions**: Ensuring that file access control (read, write, execute) remains intact. - **Handle Large Transfers Efficiently**: `rsync` is highly optimized for moving large volumes of data. - **Exclude Unwanted Folders**: The script allows you to skip specific directories (e.g., trash or temporary folders) during the transfer.

How the `moveperm.sh` Script Works

This script loops through multiple source directories, moves the files to the target directory, and then ensures the correct ownership and permissions are applied to the transferred files. Let's break it down step by step.

Step 1: Define Source and Target Directories

At the beginning of the script, we define the directories that we want to move and the target directory where the files should be transferred.

```
SOURCE_DIRS=(
    "/run/media/paulgrey/39a834f4-bb98-4978-abac-
        5980d857c556/paulgrey/"
    "/run/media/paulgrey/39a834f4-bb98-4978-abac-5980d857c556/VHD/"
)

TARGET_DIR="/run/media/paulgrey/BAKDRIVE/MAIN/DRIVE"
```

```
USER="paulgrey"
IGNORE_FOLDERS=(".lost+found" ".Trash-1000" "venv")
```

- **`SOURCE_DIRS`**: This is an array of directories where the files are currently stored. You can list as many directories as needed.
- **`TARGET_DIR`**: The destination directory where the files will be moved.
- **`USER`**: The user who should own the files after they are transferred.
- **`IGNORE_FOLDERS`**: A list of directories to exclude from the transfer (such as system folders or trash directories).

Why Use Arrays?

Using arrays for the source directories allows you to move files from multiple locations to the same target directory in one script execution. It's highly flexible if you have data spread across different directories or partitions.

Step 2: Move Files Using Rsync

The `move_contents` function handles the actual file transfer using `rsync`. This function processes each source directory, excluding any folders specified in the `IGNORE_FOLDERS` array.

```
move_contents() {
    local source_dir="$1"
    local target_dir="$2"
    local exclude_patterns=()
    for ignore in "${IGNORE_FOLDERS[@]}"; do
        exclude_patterns+=("--exclude=$ignore")
    done
    sudo rsync -avh --progress --partial --ignore-existing
        "${exclude_patterns[@]}" "$source_dir" "$target_dir"
}
```

- **`--exclude=$ignore`**: This flag ensures that files in the ignored directories are not transferred.
- **`rsync -avh --progress --partial --ignore-existing`**:
  - **`-a`**: Archive mode, which preserves symbolic links, permissions, and ownership.
  - **`-v`**: Verbose output to show the progress.
  - **`-h`**: Human-readable format for file sizes.
  - **`--progress`**: Displays the progress of the transfer.
  - **`--partial`**: Ensures that partially transferred files are saved if the transfer is interrupted.
  - **`--ignore-existing`**: Prevents overwriting existing files in the target directory unless they have changed.

This function ensures that files are moved from the source directories to the target directory while excluding unnecessary folders and preserving file attributes.

Step 3: Apply Correct Ownership and Permissions

Once the files are moved, it is essential to ensure that the correct ownership and permissions are applied to the files and directories. The `change_permissions` function takes care of this.

```
change_permissions() {
    sudo chown -R $USER:$USER "$TARGET_DIR"
    sudo find "$TARGET_DIR" -type f -exec chmod u+rw,go+r {} +
    sudo find "$TARGET_DIR" -type d -exec chmod u+rwx,go+rx {} +
}
```

- **`chown -R $USER:$USER "$TARGET_DIR"`**: Changes ownership of all files and directories in the target directory to the specified user.
- **`find "$TARGET_DIR" -type f -exec chmod u+rw,go+r {}`**: Ensures that all files in the target directory have read and write permissions for the owner and read permissions for everyone else.
- **`find "$TARGET_DIR" -type d -exec chmod u+rwx,go+rx {}`**: Ensures that all directories have read, write, and execute permissions for the owner, and read and execute permissions for everyone else.

This step ensures that the files remain accessible after they are moved and that permissions are correctly set based on your system's requirements.

Step 4: Execute the Script

The script loops through all the source directories and calls the `move_contents` function for each directory, followed by the `change_permissions` function to ensure everything is set up correctly.

```
for SOURCE_DIR in "${SOURCE_DIRS[@]}"; do
    move_contents "$SOURCE_DIR" "$TARGET_DIR"
done

change_permissions
```

This ensures that all files are moved from the defined source directories to the target directory, and ownership and permissions are correctly set.

Why This Script is Effective

1. **Preserves File Permissions**: Ensuring that permissions and ownership are preserved during the transfer is critical for maintaining system security and file access control.

2. **Efficient Data Transfer**: `rsync` is a highly efficient tool that ensures only new or modified files are transferred, saving time and reducing redundant data copying.

3. **Customizable**: You can easily modify the source and target directories, as well as add or remove ignored folders based on your needs.

4. **Handles Large Transfers**: With the ability to resume partially transferred files, this script is robust enough for large-scale backups and file transfers.

How to Use the Script

1. **Create the Script**: Copy the following code and save it as `moveperm.sh`.

```
#!/bin/bash

SOURCE_DIRS=(
    "/run/media/paulgrey/39a834f4-bb98-4978-abac-
        5980d857c556/paulgrey/"
    "/run/media/paulgrey/39a834f4-bb98-4978-abac-5980d857c556/VHD/"
)

TARGET_DIR="/run/media/paulgrey/BAKDRIVE/MAIN/DRIVE"
USER="paulgrey"
IGNORE_FOLDERS=(".lost+found" ".Trash-1000" "venv")

# Function to move contents using rsync
move_contents() {
    local source_dir="$1"
    local target_dir="$2"
    local exclude_patterns=()
    for ignore in "${IGNORE_FOLDERS[@]}"; do
        exclude_patterns+=("--exclude=$ignore")
    done
    sudo rsync -avh --progress --partial --ignore-existing
        "${exclude_patterns[@]}" "$source_dir" "$target_dir"
}

# Change permissions
change_permissions() {
    sudo chown -R $USER:$USER "$TARGET_DIR"
    sudo find "$TARGET_DIR" -type f -exec chmod u+rw,go+r {} +
    sudo find "$TARGET_DIR" -type d -exec chmod u+rwx,go+rx {} +
}

for SOURCE_DIR in "${SOURCE_DIRS[@]}"; do
    move_contents "$SOURCE_DIR" "$TARGET_DIR"
done

change_permissions
```

2. **Make the Script Executable**: Run the following command to make the script executable:

   ```
   chmod +x moveperm.sh
   ```

3. **Run the Script**: Execute the script to transfer your files and ensure permissions are correctly applied:

```
./moveperm.sh
```

This will move the files from the source directories to the target directory, while maintaining ownership and permissions.

Conclusion

Using `rsync` with the `moveperm.sh` script provides an efficient way to transfer files between directories or drives while preserving permissions and ownership. This is especially useful for backups or moving large datasets. By automating the process, you ensure that your files are transferred correctly and remain secure, without the need for manual intervention.

Feel free to adjust the script to your specific needs, whether that means adding more source directories, changing the target directory, or modifying the permissions. Let me know if you have any questions!