System Automation and Maintenance with Custom Bash Scripts

In this tutorial, we will walk through various scripts designed to automate essential system administration tasks. The goal is to keep your system clean, updated, and efficient by automating operations such as finding unused packages, installing drivers, configuring OpenSSH servers with Docker, and more.

Table of Contents

1. Fetch Configuration Files

This script archives configuration files that have been accessed in the last 90 days. It looks through all user directories and copies active config files into a temporary folder before archiving them into a ZIP file.

Script: `fetch_config.sh`

```
#!/bin/bash

# Temporary directory for storing used config files
temp_dir=$(mktemp -d)
archive_name="used_config_files.zip"

# Function to determine if a config file is in use
is_in_use() {
    local config_file=$1
    if [ -f "$config_file" ]; then
        if [ $(find "$config_file" -type f -atime -90 2>/dev/null) ];
        then
            return 0
        fi
    fi
    return 1
}

# Iterate through each user in the system
```

```
for user_home in /home/*; do
    if [ -d "$user_home" ]; then
        # Process hidden files in the user's home directory
        find "$user_home" -maxdepth 1 -name ".*" -type f | while read
        config_file; do
            if is_in_use "$config_file"; then
                dest_dir="$temp_dir${config_file#/home}"
                mkdir -p "$(dirname "$dest_dir")"
                cp "$config_file" "$dest_dir"
            fi
        done

        if [ -d "$user_home/.config" ]; then
            find "$user_home/.config" -type f | while read
        config_file; do
                if is_in_use "$config_file"; then
                    dest_dir="$temp_dir${config_file#/home}"
                    mkdir -p "$(dirname "$dest_dir")"
                    cp "$config_file" "$dest_dir"
                fi
            done
        fi
    fi
done

# Create a zip archive from the copied files
zip -r "$archive_name" "$temp_dir"/*

# Clean up the temporary directory
rm -rf "$temp_dir"

echo "Used configuration files have been archived into $archive_name."
```

This script is useful for administrators who want to periodically back up actively used configuration files.

2. Find Unused Packages

This script checks which installed packages have not been accessed in the last 30 days, making it easier to remove unnecessary packages and free up disk space.

Script: `find_unused_pkg.sh`

```
#!/bin/bash

# Find all installed packages
installed_packages=$(pacman -Qq)
```

```
# Find packages not accessed in the last 30 days
unused_packages=()
for pkg in $installed_packages; do
  pkg_dir=$(find /var/lib/pacman/local/ -maxdepth 1 -name "${pkg}-*" -
        type d)

  if [ -n "$pkg_dir" ]; then
    if ! find "$pkg_dir" -type f -atime -30 | grep -q '.'; then
      unused_packages+=("$pkg")
    fi
  fi
done

# Print unused packages
printf "%s\n" "${unused_packages[@]}"
```

This script helps in identifying packages that are no longer in use, allowing for safe cleanup.

3. Update GPG Keys

This script updates GPG keys, ensuring your system's package manager remains secure and trusted.

Script: `gpgupdate.sh`

```
#!/bin/bash

# Try manual update
echo "Trying manual update..."
sudo pacman -Sy archlinux-keyring
sudo pacman-key --refresh-keys
pacman-key --list-sigs | grep ImporteddKey

# Check if GPG database needs recreation
echo "Checking GPG database..."
ls -l /etc/pacman.d/gnupg
if [ -d /etc/pacman.d/gnupg ]; then
    echo "Backing up and recreating GPG database..."
    sudo mv /etc/pacman.d/gnupg /etc/pacman.d/gnupg.back
    sudo pacman-key --init
    sudo pacman-key --populate
    sudo pacman-key --refresh-keys
fi

# Receive a specific key
echo "Receiving specific GPG key..."
```

```
sudo gpg --keyserver pool.sks-keyservers.net --recv-keys
        33C235A34C46AA3FFB293709A328C3A2C3C45C06

# Try debug
echo "Trying debug..."
sudo gpg --debug-level guru --keyserver hkp://keys.gnupg.net --search-
        keys 13975A70E63C361C73AE69EF6EEB81F8981C74C7

echo "Script execution complete."
```

This script ensures that your system is always up-to-date with the latest keys, preventing package verification issues.

4. Install Extra Packages

This script helps install all the additional packages on your system in one go. It outputs a script that installs every package you've added beyond the base system.

Script: `install_extra_packages.sh`

```
#!/bin/bash

# List of packages to install:
sudo pacman -S --noconfirm accountsservice
sudo pacman -S --noconfirm alacritty
sudo pacman -S --noconfirm alsa-firmware
sudo pacman -S --noconfirm alsa-plugins
sudo pacman -S --noconfirm alsa-utils
...
```

This is a great script for quickly setting up a new system or re-installing packages after a fresh install.

5. Move Files with Permissions

This script automates the process of moving files from one drive to another while ensuring file permissions and ownerships are maintained.

Script: `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
```

This is particularly useful for backing up large amounts of data to an external drive while retaining the correct permissions.

6. Set Up an OpenSSH Server in Docker

This script runs an OpenSSH server within a Docker container. It uses environment variables to configure the server with specific user credentials.

Script: `openssh_docker.sh`

```
#!/bin/bash

echo "Creating docker openssh server"

docker run -d \
  --name=openssh-server \
  --hostname=paulgrey-omen \
  -e PUID=1000 \
  -e PGID=1000 \
  -e TZ=Etc/UTC \
```

```
  -e PUBLIC_KEY_FILE=/home/paulgrey/.sshk/id_rsa.pub \
  -e PUBLIC_KEY_DIR=/home/paulgrey/.sshk \
  -e SUDO_ACCESS=false \
  -e PASSWORD_ACCESS=false \
  -e USER_PASSWORD=Notnewtothegame89! \
  -e USER_NAME=paulgrey \
  -p 2222:2222 \
  -v /home/paulgrey/.sshc/sshd_config \
  --restart unless-stopped \
  lscr.io/linuxserver/openssh-server:latest

echo "Server created"
```

This is ideal for setting up a lightweight OpenSSH server with minimal configuration effort.

7. List Installed Drivers

This script lists all the installed drivers on your system using `lspci`. It also generates a script to reinstall the drivers later.

Script: `drivers.sh`

```
#!/bin/bash

# Output all installed drivers recognized by l

spci
installed_drivers=$(sudo lspci -k | awk '/Kernel driver in use:/
        {print $NF}' | sort -u)

install_script="install_drivers.sh"
echo "#!/bin/bash" > "$install_script"
echo "# Script to install currently installed drivers" >>
        "$install_script"

for driver in $installed_drivers; do
    echo "Installed driver: $driver"
    echo "sudo pacman -S --noconfirm $driver" >> "$install_script"
done

echo "Script to install currently installed drivers generated as
        $install_script."
```

This script is useful for administrators who need to track and reinstall drivers after a system rebuild.

Conclusion

These scripts are great for automating various system administration tasks. From updating packages and cleaning up unused files to installing drivers and setting up servers, these tools help streamline the maintenance process and keep your Linux system running efficiently.