Tutorial: Finding Unused Packages for System Cleanup

Over time, your system accumulates packages that may no longer be in use. These packages can take up valuable disk space without providing any real benefit. To maintain a lean, efficient system, it's a good idea to periodically check for packages that haven't been accessed recently and consider removing them.

In this tutorial, we will walk through a bash script that automates the process of finding packages that have not been accessed in the last 30 days. This script is particularly useful for system administrators and users who want to free up disk space by safely removing unused packages.

Why Should You Find and Remove Unused Packages?

As you install software, dependencies and libraries may accumulate on your system. Some of these packages become redundant over time as they are no longer required by any other software or simply aren't used anymore. Identifying and removing such packages can:

- **Free Up Disk Space**: Unused packages can take up significant space on your drive.
- **Improve Performance**: A cleaner system is often more efficient and responsive.
- **Reduce Maintenance Overhead**: Fewer packages mean fewer updates and less risk of package conflicts.

By automating the task of finding unused packages, you can regularly clean up your system with minimal effort.

How the `find_unused_pkg.sh` Script Works

This script checks all installed packages on your system and identifies those that haven't been accessed in the last 30 days. It then outputs a list of those packages, which you can review and decide to remove.

Here's a detailed explanation of how the script operates.

Step 1: Fetch All Installed Packages

The script begins by using the `pacman` package manager to list all installed packages.

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

- **`pacman -Qq`**: This command lists all installed packages in a simplified format (only package names, no additional details).

By listing all installed packages, we can loop through each one to check whether it's still being used.

Step 2: Check the Access Time of Each Package

Next, the script checks the `/var/lib/pacman/local/` directory, which stores metadata for each

installed package. For each package, it looks at the access time of its files to determine whether the package has been used in the last 30 days.

```
# 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
```

- **`find /var/lib/pacman/local/ -maxdepth 1 -name "${pkg}-*" -type d`**: This searches for the directory associated with each package in the Pacman local database.
- **`find "$pkg_dir" -type f -atime -30`**: This checks the files in the package directory and looks for those that have been accessed in the last 30 days.
- **`-atime -30`**: The `-atime` flag stands for "access time." The `-30` value specifies files that haven't been accessed for 30 days or more.

If no files have been accessed in the last 30 days, the package is considered unused and is added to the `unused_packages` array.

Step 3: Output Unused Packages

Once all packages have been evaluated, the script prints a list of the unused packages.

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

This gives you a clear, readable list of packages that have not been accessed in the last 30 days. From here, you can decide which ones to remove.

Why This Script is Effective

1. **Automated Cleanup**: Manually tracking when a package was last used is cumbersome. This script automates that process, saving you time and effort.

2. **Efficient Disk Usage**: By identifying packages that are no longer used, you can clean up your system and free up disk space, improving overall system performance.
3. **Customizable**: The script is easy to modify if you want to change the time period (e.g., 60 days instead of 30) or adjust the scope of packages being checked.

How to Use the Script

1. **Create the Script**: Copy the script below and save it as `find_unused_pkg.sh`.

```bash
#!/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[@]}"
```

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

   ```
   chmod +x find_unused_pkg.sh
   ```

3. **Run the Script**: Execute the script to find unused packages:

   ```
   ./find_unused_pkg.sh
   ```

   After the script runs, you will see a list of packages that have not been accessed in the last 30 days. Review this list and remove packages you no longer need using Pacman:

   ```
   sudo pacman -Rns <package_name>
   ```

Conclusion

By regularly running this script, you can keep your system clean and efficient. Unused packages, especially on systems that undergo frequent software installations and updates, can take up unnecessary space and contribute to system clutter. This script helps identify packages that haven’t been accessed in a while, making it easier for you to manage and maintain your system.

Feel free to customize the script based on your needs. For example, you could adjust the time window by changing `-atime -30` to a different number of days, or even automate this task with a cron job to regularly clean your system.