Date: 12.06.2025

Linux Utilities

Contents

1. System Backup and restore

a. Using Timeshift

- Timeshift is for backing up system files and settings.
- So when you are configuring your system and making some customization and mess the system up in the process, you could revert to the older system snapshot.
- Timeshift is designed to protect system files and settings. It is NOT a backup tool and is not meant to protect user data. Entire contents of users' home directories are excluded by default.

❖ **Installing Timeshift**

i. For Ubuntu and Linux Mint: `sudo apt install timeshift`
ii. For Arch Linux and its derivatives: `yay –S timeshift`
iii. For Fedora and its derivatives: `sudo dnf install timeshift`

❖ **Creating a system backup**

i. Open TimeShift
   Launch TimeShift from system menu. It will now ask for your user account password
ii. Select Snapshot Type
   Two options to select snapshot type:
   - RSync – Snapshots are taken using rsync and hard links. Common files are shared between snapshots thus saving disk space. Each snapshot is a full system backup that can be browsed with a file manager.
   - BTRFS – Snapshots are taken using the in-built features of the BTRFS file system. BTRFS Snapshots are supported only on BTRFS system.
iii. Select the storage device
   Select location for backup storage.
   - If computer’s hard disk is used, Timeshift backups can be created automatically on schedule
   - If external USB disk is used, manual backups must be created when plugged in USB disk, unless it is plugged in always.

   Ensure that the used device is formatted to a Linux Filesystem.
iv. Configure Scheduled Snapshots
   - Select from several options – monthly, weekly, daily, hourly,weekly
   - You can set the no of copies you want to keep, of each type of scheduled backup.
v. Select the files / folders
   - Select the parts of your system for back up.
   - By default, the user files are not backed up. Only those files that are needed to make the system up and running, are backed up.
   - If you back up the user files, while restoring, those files will be overwritten, and any changes you have made after creating the backup will be gone.
   - There is also an option to backup hidden files

   Click Finish.
vi. Create the backup
   - Once initial setup is completed, TimeShift homepage will be opened.
   - Click Create to create the first backup.

- **Restoring a system backup**
  - From the same OS
    1. Launch TimeShift
    2. Click Restore option
    3. Select a Restore Image
    4. Hit Restore
  - Restoring when you can't log into Linux System
    1. Use a live Linux USB.
    2. Boot into Linux Live Session.
    3. Install TimeShift
       `sudo add-apt-repository universe`
       `sudo apt install timeshift`
    4. Go through the setup wizard.
    5. Select the backup drive previously used.
    6. Select the Backup you require
    7. Hit Restore
    8. Provide restore paths according to your system.
    9. Recommended to let TimeShift install the bootloader again.
    10. Once Next is pressed, a dry-run is performed.
    11. Confirm actions you approve.
    12. Restoration will start once you hit Next
    13. Reboot the system and remove the Live USB. Then boot into the restored Linux system.

b. Using tar

- This Unix-like command creates and manipulates file archives.
- **Tape archive** is used for backups, compression, and directory archiving.
- Tar compresses several files and directories into a single archive file.
- tar supports It supports compression methods like gzip and bzip2.

- **Creating Backup**

  tar -cvzf path/to/backup/home_backup.tar.gz /home/username

  - -c: creates a new archive
  - -v: shows the list of files being processed (verbose mode)
  - -z: compresses the archive using gzip for space efficiency
  - -f: specifies the archive filename (always the last argument)

- **Restoring Backup**

  tar -xvzf path/to/backup/home_backup.tar.gz -C /path/to/restore

  - -x: extracts files from the the archive
  - -C: changes the extraction directory (to avoid overwriting existing files)

2. Log Management and Analysis

- Logs are records of events that occur in the system.
- Created by the operating system, applications, and services.
- Useful for:
    - Diagnosing issues.
    - Auditing security events.
    - Monitoring system behavior.

❖ **Stages in Log Management**

i. **Log Generation:**
   - Logs are automatically created by system processes and applications.

ii. **Log Collection:**
   - Tools like rsyslog collect and store logs centrally.

iii. **Log Rotation:**
   - Prevents logs from growing too large.
   - Managed using logrotate.

iv. **Log Archiving:**
   - Older logs are compressed and stored for future reference.

v. **Log Analysis:**
   - Analyzing logs to identify trends, errors, or security breaches.
   - Tools: grep, awk, or advanced tools like ELK Stack.

❖ **Types of Logs in Linux**

System	• Capture general system activity, errors, warnings, and status messages. • Provide insights into the functioning of the operating system and its components. • Example: `/var/log/syslog` (in Debian/Ubuntu)
Authentication	• Track authentication-related events, such as user logins, failed login attempts, and privilege escalations. • Essential for auditing security incidents and detecting unauthorized access. • Example: `/var/log/auth.log` (in Debian/Ubuntu)
Application	• Logs generated by specific applications for debugging, usage tracking, and error reporting. • Help troubleshoot issues with individual software or services. • Example: `/var/log/apache2/access.log` (Apache Web Server)
Boot	• Record information about the system boot process, including kernel initialization, services startup, and hardware detection. • Useful for diagnosing boot-time errors or delays. • Example: `/var/log/boot.log` (in Debian/Ubuntu)
Kernel	• Contain messages generated by the Linux kernel. • Track hardware-related activities, driver issues, and kernel-level errors. • Example: `/var/log/kern.log` (in Debian/Ubuntu)

- **Accessing and Managing Logs**

   i. **Using `cat` command**
      - The CAT command in Linux is used to concatenate and display the contents of log files.
      - Opens the Log File that we want.
      - `cat /var/log/auth.log`

   ii. **Using `grep` command**
      - The GREP Command filters and extracts specific information from Linux logs.
      - Prompts the **Log Files to Manage on Linux** those will have the proper string in the name.
      - `grep “invalid” /var/log/auth.log`

   iii. **Using `sort` command**
      - The log files will now have all the information stored in ascending order.
      - `sort /var/log/auth.log`

   iv. **Using `uniq` command**
      - Promote the messages that are **Unique** & make suggested changes in the file.
      - `uniq /var/log/auth.log`

   v. **Using `journalctl` command**
      - `journalctl –b` shows system boot logs
      - Has many other options for accessing and managing logs

3. Kernel Modules and Drivers

❖ **Kernel Drivers**

- Translators between the operating system (OS) and the physical devices connected to our computers
- Ensure the smooth functionality of our Linux systems.
- The kernel implements kernel drivers, which reside in the kernel's source code and load into the memory during system boot.
- Operating at the kernel level, kernel drivers have direct access to the hardware, facilitating efficient and secure communication with devices.\
- Example: nvidia

❖ **Kernel Modules**

- Kernel modules are separate pieces of code that can dynamically load and unload from the kernel during runtime.
- Extend the kernel's functionality by adding or modifying device drivers or other kernel components without requiring a complete kernel recompilation.
- This dynamic nature makes them particularly useful when working with hardware that may be frequently added or removed.
- Examples: nvidia, snd_hda_intel, iwlwifi, btusb, usb_storage

Feature	Kernel Module	Kernel Driver
Definition	Loadable code extending kernel at runtime	Program enabling OS-hardware communication
Loading	Dynamically loaded/unloaded	Loaded at boot or as a module
Scope	Can be a driver or other kernel extension	Always provides hardware interface
Flexibility	High (dynamic, modular)	Less flexible if built-in, more if modular
Typical Use	Device drivers, filesystems, protocol stacks, etc.	Device management (network, sound, graphics)
Example	nvidiafbmodule for graphics	nvidiadriver for NVIDIA GPU

❖ **Listing available Kernel Modules:** `ls /lib/modules/$(uname -r)`

❖ **Listing available Kernel Drivers:** `ls /lib/modules/$(uname -r)/kernel/drivers/`

❖ **List all modules:** `lsmod`

❖ **Loading a Kernel Module**

`sudo modprob my_module`

- Execute `modprobe` command with administrative privileges (sudo) followed by the module's filename, typically without the .ko extension.
- This loads the specified module and also resolves and loads any dependencies required by that module.
- Loads the specified kernel module into the kernel's address space and ensures its proper functioning within the kernel environment

- **Unloading a Kernel Module**

  `sudo modprob –r my_module`
  - Run `modprobe` with the -r option.
  - Unloads the loaded kernel module from the running kernel

- **`lpsci` command**

  `lpsci –k`
  - Linux utility that provides information about the Peripheral Component Interconnect (PCI) devices connected to our system.
  - Provides detailed information about the hardware devices and their corresponding drivers.
  - When used with the -k flag, it displays details about the PCI devices and the associated kernel driver and kernel modules for those devices.

Transforming Enterprises for Future

4. Resources

- Guide to Backup and Restore Linux Systems with Timeshift
- Linux Backup and Restore Commands Cheat Sheet - Scaler Topics
- Backup and Restore in Linux
- Monitoring Linux And Log Management in Linux
- How to Manage Logs in Linux? - GeeksforGeeks
- What’s the Difference Between Kernel Drivers and Kernel Modules? | Baeldung
- Linux Device Drivers Tutorial | Linux Drivers and Kernel Modules
- How to Add, Remove, and Manage Linux Kernel Modules (Drivers) - nixCraft