LINUX PROGRAMMING: ASSIGNMENT 10

Networking:

hostnamectl

view or set hostname and related settings.

The hostnamectl command in Linux is used to query and change the system hostname and related settings. It's part of the systemd suite and works on systems that use systemd.

What it does:

 Displays current hostname information

* Sets static, transient, or pretty hostnames
* Shows chassis and deployment environment info

Usage:

hostnamectl

This shows details like:

* Static hostname
* Pretty hostname
* Transient hostname
* Operating system
* Kernel version
* Architecture
* Machine ID
* Boot ID

How it works:

**Manages Three Types of Hostnames:**

**.** **Static hostname** is the default and is used by most services.

* **Transient hostname** is often set by the network (e.g., DHCP) and overrides the static hostname during runtime.
* **Pretty hostname** is optional and used for display in graphical interfaces.

**No Need to Manually Edit Files:**

* Traditionally, changing a hostname required editing /etc/hostname, /etc/hosts, and rebooting.
* hostnamectl automates this by updating the necessary files and notifying the system in real time.

**System Metadata Management**

Besides hostnames, hostnamectl can also set:

* **Chassis type** (e.g., laptop, server)
* **Icon name** (used by desktop environments)
* These are stored in /etc/machine-info.

Command output:

hostnamectl

Static hostname: myserver

Pretty hostname: My Server

Transient hostname: tempserver

Icon name: computer-laptop

Chassis: laptop

Machine ID: 1234567890abcdef1234567890abcdef

Boot ID: abcdef1234567890abcdef1234567890

Operating System: Ubuntu 22.04.3 LTS

Kernel: Linux 6.2.0-33-generic

Architecture: x86-64

**hostnamectl option flags:**

Here are the most commonly used options:

|  |  |  |
| --- | --- | --- |
| --static |  | sudo hostnamectl set-hostname myhost --static |
| --transient |  | sudo hostnamectl set-hostname temp-host --transient |
| --pretty |  | sudo hostnamectl set-hostname "My Laptop" --pretty |
| --help |  | hostnamectl --help |
| status |  | hostnamectl status |
| set-hostname [name] |  | sudo hostnamectl set-hostname server01 |

**history of hostnamectl:**

Origin

* Introduced by systemd: The hostnamectl command was created as part of the systemd-hostnamed service, which is a component of the broader systemd init system.
* First appeared: Around 2011–2012, when systemd began gaining traction as a replacement for traditional init systems like SysVinit and Upstart.

**Purpose of Creation:**

To **standardize hostname management** across Linux distributions.

* To **replace manual editing** of files like /etc/hostname, /etc/hosts, and /etc/machine-info.
* To support **multiple types of hostnames**:
* **Static**: Persistent across reboots
* **Transient**: Temporary, often set by DHCP or mDNS
* **Pretty**: Human-readable display name

|  |  |
| --- | --- |
|  |  |

**Replaces Older Methods:**

Before hostnamectl, hostnames were managed by:

* Editing /etc/hostname manually
* Using the legacy hostname command (still available but limited)
* Modifying /etc/hosts for name resolution.

**curl**

transfer data from/to a URL

The curl command in Linux is used to transfer data to or from a server using various internet protocols like HTTP, HTTPS, FTP, and more.

What is curl

* **Purpose**: curl stands for *Client URL*. It's a command-line tool used to interact with servers via URLs.
* **Protocols Supported**: HTTP, HTTPS, FTP, FTPS, SCP, SFTP, LDAP, MQTT, SMTP, and many others.

Usage:

* Downloading files
* Uploading data
* Testing REST APIs
* Checking website availability

How it works:

* **Clear Purpose**: You defined curl as a tool for transferring data using URLs—perfect.
* **Syntax and Examples**: The examples are practical and varied (GET, POST, headers, redirects).
* **Advanced Options**: Including authentication and proxy usage shows depth.

Command output:

**Download a File**

curl -O https://example.com/file.txt

* **Explanation**: Downloads the file and saves it with its original name (file.txt).
* **Output**:

% Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed 100 1270 100 1270 0 0 1270 0 --:--:-- --:--:-- --:--:-- 1270

2. **Send a POST Request**

curl -X POST -d "username=shweta&password=1234" https://example.com/login

* **Explanation**: Sends form data using POST method.
* **Output** (depends on server response):

{ "status": "success", "message": "Logged in successfully" }

3. **View Response Headers Only**

curl -I https://example.com

* **Explanation**: Fetches only the HTTP headers.
* **Output**:

HTTP/1.1 200 OK Date: Mon, 20 Oct 2025 16:50:00 GMT Content-Type: text/html Content-Length: 3056 Connection: keep-alive

4. **Follow Redirects**

curl -L https://example.com

* **Explanation**: Follows HTTP 3xx redirects until the final destination.
* **Output**: Final page content or headers depending on additional flags.

5. **Basic Authentication**

curl -u user:pass https://example.com/protected

* **Explanation**: Sends credentials using HTTP Basic Auth.
* **Output**:

Welcome, user!

6. **Save Output to a File**

curl https://example.com -o homepage.html

* **Explanation**: Saves the response body to homepage.html.
* **Output**: No terminal output unless -v or --verbose is used.

Would you like this formatted as a printable reference sheet or turned into flashcards for your Linux programming study set?

**Option flags:**

curl -O <https://example.com/file.txt>

* **flag: -O**
* **Purpose**: Saves the file using its original name from the URL.
* **Output**: Progress bar showing download percentage.

curl -o homepage.html <https://example.com>

* **Flag**: -o [filename]
* **Purpose**: Saves response body to a specified file.
* **Output**: Silent unless combined with -v.

curl -I <https://example.com>

* **Flag**: -I
* **Purpose**: Sends a HEAD request to fetch headers only.
* **Output**:

HTTP/1.1 200 OK

Content-Type: text/html

curl -L <https://example.com>

* **Flag**: -L
* **Purpose**: Automatically follows HTTP 3xx redirects.
* **Output**: Final page content after redirection.

curl -u user:pass <https://example.com/protected>

* **Flag**: -u [username:password]
* **Purpose**: Sends credentials using HTTP Basic Auth.
* **Output**: Protected content if credentials are valid.

curl --limit-rate 100K <https://example.com>

* **Flag**: --limit-rate [speed]
* **Purpose**: Throttles download speed.
* **Output**: Slower transfer with progress bar.

curl -x proxy.example.com:8080 https://example.com

* **Flag**: -x [proxy]
* **Purpose**: Routes request through a proxy.
* **Output**: Same as direct request, but via proxy.

**History of curl**:

**Origin:**

* **1996**: Brazilian developer Rafael Sagula released HttpGet version 0.1 on November 11, 1996. It was a simple command-line tool for fetching web content.
* **Daniel Stenberg** discovered HttpGet while building an IRC bot that needed to fetch currency exchange data from the web. He modified and extended the tool, releasing version 0.2 on December 17, 1996.
* **1997**: Daniel became the maintainer and continued developing it. By April 1997, HttpGet supported HTTP proxies and accepted URLs directly on the command line.

**Transition to curl:**

* **1998**: Daniel renamed the project to **curl**, short for *Client for URLs*. The name reflected its broader capabilities beyond just HTTP.
* The first official release of curl under its new name was on **March 20, 1998**, marking the beginning of its modern evolution.

About the Creator:

* **Daniel Stenberg** is a Swedish software engineer who started curl while working at Frontec Tekniksystem. He’s still the lead developer and a prominent figure in open-source networking tools

Key Features Over Time

* **Protocol Expansion**: Originally supporting HTTP, curl now handles over 25 protocols including HTTPS, FTP, FTPS, SCP, SFTP, LDAP, MQTT, and more.
* **libcurl**: Alongside the command-line tool, Daniel developed libcurl, a library that allows developers to integrate URL transfer capabilities into their applications.
* **Cross-platform**: Written in C, curl runs on over 29 platforms including Linux, Windows, macOS, and embedded systems.
* **Open Source**: Licensed under a permissive curl license (inspired by MIT), it has attracted a large contributor base and remains actively maintained.