**CURL Command in Linux with Examples**

* curl is a command-line utility for transferring data to or from a server, employing a range of internet protocols such as HTTP, HTTPS, FTP, SCP, and SFTP.

Whether you want to download a file, test a REST API, or simply verify that a website is up and running, curl is your best thing to do. It is accessed directly from the terminal no need to fire up a browser or install some slick app.

**Syntax of curl Command**

curl [options] [URL]

* [**options**]: Can be various command-line flags that modify the behavior of curl
* [**URL**]: Specifies the location from which to fetch or send data.

## Navigation and File Listing

### cd ~/Documents/demo\_linux/build/libs

**Purpose**: Change directory to the specified path **Explanation**:

* cd stands for "change directory"
* ~ represents the user's home directory
* This navigates to the libs folder containing Java JAR files

### ll

**Purpose**: List files in long format (alias for ls -l) **Explanation**:

* Shows detailed file information including permissions, ownership, size, and timestamps
* Output shows JAR files, log files, and other application-related files
* File sizes are displayed in bytes (e.g., 26876088 bytes for the main JAR)

## Process Management

### nohup java -jar demo\_linux-0.0.1-SNAPSHOT.jar >output.log 2>&1 &

**Purpose**: Run Java application in background, immune to hangups **Explanation**:

* nohup prevents the process from being terminated when terminal closes
* java -jar executes the JAR file
* >output.log redirects standard output to a log file
* 2>&1 redirects standard error to the same location as standard output
* & runs the command in background
* Returns process ID [1] 3837

### ps aux | grep demo

**Purpose**: Find running processes related to "demo" **Explanation**:

* ps aux lists all running processes
* grep demo filters results containing "demo"
* Shows the Java process consuming 67.2% CPU and 8.2% memory
* PID 3837 matches the background process started earlier

## Log File Operations

### tail -1000 nohup.out

**Purpose**: Display last 1000 lines of the nohup output file **Explanation**:

* tail shows the end of files
* -1000 specifies number of lines to display
* Shows Spring Boot startup logs and application lifecycle

### cat nohup.out

**Purpose**: Display entire contents of the nohup output file **Explanation**:

* cat concatenates and displays file contents
* Shows complete application logs including startup sequence and shutdown

### tail -f output.log

**Purpose**: Follow log file in real-time **Explanation**:

* -f flag makes tail "follow" the file
* Shows new lines as they're added to the file
* Useful for monitoring live application logs
* Requires Ctrl+C to exit

## Network Monitoring

### sudo lsof -i -Pn

**Purpose**: List open files and network connections **Explanation**:

* lsof lists open files (including network sockets)
* -i shows internet connections
* -P shows port numbers instead of service names
* -n shows IP addresses instead of hostnames
* Requires sudo for complete system information
* Shows services like SSH (port 22), MySQL (port 3306), and our Java app (port 8080)

### sudo netstat -tulnp

**Purpose**: Display network connections and listening ports **Explanation**:

* netstat shows network statistics
* -t shows TCP connections
* -u shows UDP connections
* -l shows only listening ports
* -n shows numerical addresses
* -p shows process IDs and names
* Alternative to lsof for network monitoring

## File Content Operations

### cat output.log

**Purpose**: Display the application log file contents **Explanation**:

* Shows Spring Boot application startup sequence
* Includes Tomcat server initialization on port 8080
* Shows servlet initialization when first request is made

## HTTP Testing with cURL

**Note**: The following HTTP request testing examples use https://httpbin.org — a public API specifically designed for testing HTTP requests. It's an excellent service that echoes back request details, making it perfect for learning and debugging HTTP interactions.

### curl http://localhost:8080/api/hello

**Purpose**: Test API endpoint with GET request **Explanation**:

* curl is a command-line HTTP client
* Makes GET request to Spring Boot application
* Returns "Hello from Spring Boot!" response

### curl "http://localhost:8080/api/greet?name=Alex"

**Purpose**: Test API endpoint with query parameter **Explanation**:

* Quotes protect URL from shell interpretation
* ?name=Alex adds query parameter
* Returns personalized greeting "Greetings, Alex!"

### curl http://localhost:8000 -o test.html

**Purpose**: Attempt to download content and save to file **Explanation**:

* -o test.html saves response to specified file
* Fails because port 8000 is not running any service
* Shows connection error after 3ms timeout

### curl http://localhost:8080 -o test.html

**Purpose**: Download content from working port **Explanation**:

* Successfully connects to port 8080
* Downloads 93 bytes of content
* Saves response to test.html file

### curl -I http://localhost:8080

**Purpose**: Send HEAD request to get only headers **Explanation**:

* -I flag sends HEAD request instead of GET
* Shows HTTP status (204) and response headers
* Useful for checking server status without downloading content

### curl -v http://localhost:8080

**Purpose**: Verbose output showing full HTTP transaction **Explanation**:

* -v enables verbose mode
* Shows DNS resolution, connection details, and full HTTP exchange
* Returns JSON response with HAL (Hypertext Application Language) format
* Useful for debugging HTTP communications

### curl -O https://httpbin.org/image/png

**Purpose**: Download file and save with original filename **Explanation**:

* -O saves file with same name as in URL (becomes "png")
* Downloads 8090 bytes from httpbin.org test service
* Shows download progress with transfer statistics
* httpbin.org provides various test endpoints including sample images

### curl -X POST -d "username=user&password=pass" https://httpbin.org/post

**Purpose**: Send POST request with form data **Explanation**:

* -X POST specifies HTTP POST method
* -d sends data in request body
* Sends form-encoded data (application/x-www-form-urlencoded)
* httpbin.org echoes back the request details, making it perfect for testing form submissions

### curl -X POST -H "Content-Type: application/json" -d '{"name":"Rashmika"}' https://httpbin.org/post

**Purpose**: Send POST request with JSON data **Explanation**:

* -H adds custom header specifying JSON content type
* -d sends JSON payload in request body
* Server parses JSON and includes it in response under "json" field
* httpbin.org's /post endpoint is ideal for testing REST API JSON interactions

### curl -H "Authorization: Bearer YOUR\_TOKEN" https://httpbin.org/get

**Purpose**: Send authenticated request with bearer token **Explanation**:

* -H adds Authorization header
* "Bearer YOUR\_TOKEN" is common authentication pattern
* Shows how to include authentication in API requests
* httpbin.org's /get endpoint echoes headers back, perfect for verifying authentication headers

### curl https://httpbin.org/ip

**Purpose**: Get public IP address **Explanation**:

* Simple endpoint that returns client's public IP address
* Shows "112.134.208.208" as the external IP address
* httpbin.org's /ip endpoint is useful for testing network connectivity and NAT detection

### curl https://httpbin.org/anything

**Purpose**: Test endpoint that accepts any HTTP method **Explanation**:

* Returns complete request information regardless of HTTP method used
* Shows method (GET), headers, and all request details
* httpbin.org's /anything endpoint is a useful catch-all for testing various HTTP scenarios

### curl -v https://httpbin.org/get

**Purpose**: Verbose HTTPS connection showing SSL/TLS details **Explanation**:

* Shows complete SSL/TLS handshake process
* Displays certificate information and verification
* Uses HTTP/2 protocol (shown in connection details)
* Demonstrates secure connection establishment process
* httpbin.org's /get endpoint with verbose output is excellent for debugging HTTPS connections

### curl -o hello.zip ftp://speedtest.tele2.net/1MB.zip

**Purpose**: Attempt FTP download (failed) **Explanation**:

* Tries to download file via FTP protocol
* -o hello.zip would save as local file
* Fails to connect to FTP server (port 21)
* Shows curl supports multiple protocols beyond HTTP

## Directory Listing

### ll (final command)

**Purpose**: List current directory contents **Explanation**:

* Shows various directories (Desktop, Documents, Downloads, etc.)
* Includes files created during the session (nohup.out, nohup.txt)
* File permissions shown as drwxr-xr-x for directories, -rw-rw-r-- for files
* Total size shows 60 KB for the directory

## Key Learning Points

1. **Process Management**: Using nohup and & to run background processes
2. **Log Monitoring**: Using tail -f for real-time log following
3. **Network Debugging**: Using lsof and netstat to check open ports
4. **HTTP Testing**: Using curl for comprehensive API testing
5. **File Operations**: Redirecting output and managing log files
6. **System Monitoring**: Checking running processes and resource usage

This session demonstrates a complete workflow for deploying, monitoring, and testing a web application using command-line tools.