Advanced Linux Commands

---

1. curl

Explanation

`curl` is used to **fetch data from a URL** (like a webpage or API) from the terminal. It's a powerful tool for downloading files, testing APIs, and interacting with web servers without opening a browser.

Syntax

Basic usage:
curl URL

Save output to file:
curl -o filename URL

Download with original filename:
curl -O URL

Examples

- Show homepage HTML of example.com in terminal:
  curl https://example.com
- Download a file and save as `page.html`:
  curl -o page.html https://example.com
- Download a file with original name:
  curl -O https://example.com/index.html
- Fetch only headers (useful for checking HTTP status):
  curl -I https://example.com

Exercises

1. Use `curl` to fetch `https://example.com` and view output in the terminal.
2. Save `https://example.com` to a file named `example.html`.
3. Try a URL that does not exist and observe the error message.
4. If you know any public API, call it with `curl URL` and examine the JSON output.
5. Use `curl -I` to fetch headers only from a website.
6. Download a file from the internet using `curl -O`.

---

2. awk

Explanation

awk is a **text-processing** tool used to read files line by line, split lines into columns (fields), and print or process selected fields. It's extremely useful for extracting specific data from structured text files.

Syntax

Basic format:
awk 'pattern { action }' filename

Print all lines:
awk '{print}' filename

Use field variables:
awk '{print $1, $2}' filename

Here $1 means the first column (field), $2 the second column, and $0 the entire line.

Examples

Assume file data.txt contains:
Alice 20
Bob 25
Carol 30

- Print whole lines:
  awk '{print}' data.txt
- Print only first column (names):
  awk '{print $1}' data.txt
  Output: Alice, Bob, Carol
- Print lines where second column (age) is greater than 22:
  awk '$2 > 22 {print $1, $2}' data.txt
  Output: Bob 25, Carol 30
- Count total lines:
  awk 'END {print NR}' data.txt
  Output: 3

Exercises

1. Create a file students.txt with lines like:
   Name Marks
   John 85
   Sarah 92
   Mike 78
2. Use awk to:
   - Print only student names.
   - Print only marks.
   - Print names of students with marks greater than 85.
   - Count total number of students.

3. Create a CSV file people.csv like:
   name,city,age
   John,NYC,25
   Sarah,Boston,30
   Print name and city using:
   awk -F, '{print $1, $2}' people.csv
4. Extract username from /etc/passwd file (first field separated by colon):
   awk -F: '{print $1}' /etc/passwd

---

3. sed

Explanation

sed is a **stream editor** used to search, replace, insert, or delete text in a file or input without opening an editor. It processes text line by line and outputs the result to stdout.

Syntax

General format:
sed [OPTIONS] 'command' filename

Search and replace (first match per line):
sed 's/old/new/' filename

Search and replace (all matches per line):
sed 's/old/new/g' filename

Delete lines matching a pattern:
sed '/pattern/d' filename

Examples

Assume file.txt contains:
I like apples.
Apples are tasty.
I eat apples daily.

- Replace first apples with oranges in each line:
  sed 's/apples/oranges/' file.txt
  Output:
  I like oranges.
  Apples are tasty.
  I eat oranges daily.
- Replace all occurrences of apples with oranges (case-insensitive):
  sed 's/apples/oranges/gi' file.txt
- Delete all lines containing the word tasty:
  sed '/tasty/d' file.txt
- Save changes to the original file (use -i flag):
  sed -i 's/apples/oranges/g' file.txt

Exercises

1. Create fruits.txt containing:
   apple pie
   apple juice
   banana bread
   apple sauce
2. Use sed to:
   - Replace apple with mango only once per line.
   - Replace all apple with mango in each line.
   - Delete lines that contain the word banana.
   - Save all changes to a new file fruits_modified.txt.
3. Pipe echo into sed to replace text:
   echo "hello world" | sed 's/world/universe/'
4. Delete lines from a file that match a specific pattern and save to a new file:
   sed '/pattern/d' input.txt > output.txt

---

4. cron

Explanation

cron is a **job scheduler** that runs commands automatically at fixed times or intervals. It runs in the background (as a daemon) and is useful for automating repetitive tasks like backups, log rotation, and system maintenance.

Syntax (crontab line format)

One cron line looks like:
MIN HOUR DOM MON DOW command

Field breakdown:

- **MIN**: Minutes (0–59)
- **HOUR**: Hours (0–23, where 0 = midnight)
- **DOM**: Day of month (1–31)
- **MON**: Month (1–12, where 1 = January)
- **DOW**: Day of week (0–7, where 0 and 7 = Sunday)

Special characters:

- * = every value in that field
- , = specific values
- - = range
- / = step values

Examples

- Run backup.sh every day at 2:30 AM:
  30 2 * * * /home/user/backup.sh
- Run every minute (for testing):

-
  -
    - echo "Hello" >> /home/user/cron.log

- Run every Monday at 9:00 AM:
  ```
  0 9 * * 1 /home/user/weekly_task.sh
  ```
- Run every 15 minutes:
  ```
  */15 * * * * /home/user/script.sh
  ```
- Run on the 1st day of every month at midnight:
  ```
  0 0 1 * * /home/user/monthly_report.sh
  ```

Exercises

1. Open your crontab editor:
   ```
   crontab -e
   ```
2. Add a job that:
   - Writes current date/time to `~/time.log` every minute.
     -
       -
         -
           -
             - date >> ~/time.log
   - Runs a simple script (`~/test.sh`) every day at 10:00 AM.
     ```
     0 10 * * * ~/test.sh
     ```
   - Creates a backup every Sunday at 3:00 AM.
     ```
     0 3 * * 0 ~/backup.sh
     ```
3. List current cron jobs:
   ```
   crontab -l
   ```
4. Remove all cron jobs:
   ```
   crontab -r
   ```
5. View system logs to see cron executions:
   ```
   grep CRON /var/log/syslog
   ```

---

5. lynx

Explanation

`lynx` is a **text-based web browser** that runs inside the terminal. It's useful on servers without a GUI, for accessing websites over slow connections, or when you want to browse the web entirely from the command line.

Installation

For Debian/Ubuntu systems:
```
sudo apt install lynx
```

Syntax

Basic usage:
lynx URL

Open lynx browser (then type URL inside):
lynx

Examples

- Open example.com in text mode:
  lynx https://example.com
- Navigate within lynx:
  - Arrow keys: Move between links
  - Enter: Follow a link
  - Q: Quit the browser
  - H: Help menu

Exercises

1. Install `lynx` on your system (if not already installed).
2. Use `lynx https://example.com` and:
   - Navigate links with arrow keys.
   - Follow a link and go back.
   - Try opening any search engine (e.g., `lynx https://duckduckgo.com`).
   - Exit the browser using Q.
3. Check the difference between:
   curl https://example.com
   (shows raw HTML) vs.
   lynx https://example.com
   (displays formatted text with interactive navigation)
4. Use lynx to read a forum or text-heavy website and explore navigation.

---

6. Pipelining

Explanation

A **pipe** (`|`) sends the **output of one command** as the **input to another command**. This allows you to chain commands together to process data step by step. Only `stdout` (standard output) is passed through the pipe; errors (`stderr`) are not piped by default.

Syntax

General format:
command1 | command2 | command3

Examples

- Show only first 5 lines of ls -l output:
  ls -l | head -n 5
- Count number of lines in file.txt that contain error:
  grep "error" file.txt | wc -l
- Combine with awk:
  cat data.txt | awk '{print $1}'
- Find and count processes:
  ps aux | grep bash | wc -l
- List files, filter for .txt, and count:
  ls | grep ".txt" | wc -l

Exercises

1. Use pipes to:
   - List all files and filter only .txt files:
     ls -la | grep ".txt"
   - See running bash processes:
     ps aux | grep bash
   - Count how many .log files exist in a directory:
     ls | grep ".log" | wc -l
   - Combine grep and wc to count lines:
     grep "pattern" file.txt | wc -l
2. Create a file with multiple lines and use pipes to:
   - Display only lines containing a specific word.
   - Count total lines.
   - Sort and display unique entries.
3. Chain three commands:
   cat file.txt | grep "search_term" | sort

---

7. Redirection

Explanation

Redirection sends command **output to a file** or reads **input from a file** instead of displaying on the terminal. This is useful for saving logs, storing command results, and automating workflows.

Syntax

Output redirection (overwrite file):
command > file

Append output to end of file:
command >> file

Input redirection:
command < file

Redirect both output and error:
command > output.txt 2> error.txt

Examples

- Save `ls` output to `list.txt` (overwrites file):
  ls > list.txt
- Append `date` output to `log.txt`:
  date >> log.txt
- Run `wc` on content from `input.txt`:
  wc < input.txt
- Redirect both stdout and stderr separately:
  ls no_such_file > output.txt 2> error.txt
  (output.txt will be empty, error.txt will contain the error)

Exercises

1. Run and observe:
   - `echo "Hello" > test.txt` then `cat test.txt`
   - `echo "World" >> test.txt` then `cat test.txt` again
2. Redirect errors and output separately:
   ls no_such_file > output.txt 2> error.txt
   Inspect both files to see how output and errors are separated.
3. Combine redirection with pipes:
   ls -l | head -n 3 > top3.txt
4. Append multiple commands to a log file:
   date >> session.log
   echo "Starting backup..." >> session.log
   ls >> session.log
5. Use input redirection with text processing:
   grep "search_term" < input.txt

Summary Table

Command	Purpose	Basic Syntax
curl	Fetch data from URLs	curl URL
awk	Text processing & field extraction	awk 'pattern {action}' file
sed	Stream editing (search/replace)	sed 's/old/new/g' file
cron	Schedule automated tasks	MIN HOUR DOM MON DOW command
lynx	Text-based web browser	lynx URL
Pipelining	Chain commands	cmd1 \\| cmd2 \\| cmd3
Redirection	Save output to files	cmd > file or cmd >> file

Tips for Learning

1. **Start simple**: Practice each command individually before combining them.
2. **Experiment safely**: Use test files and directories so you don't accidentally modify important data.
3. **Use man pages**: Type `man command_name` to read detailed documentation (e.g., `man curl`, `man sed`).
4. **Combine tools**: Practice piping and redirection to see the real power of Linux commands.
5. **Keep a reference sheet**: Bookmark common syntax patterns for quick lookup.

Next Steps

- Practice these commands daily with different files and data.
- Explore the `-h` or `--help` flags on each command for additional options.
- Try writing simple shell scripts that use these commands together.
- Join Linux communities to share scripts and learn from others.

Happy learning!