Command-Line & Shell Cheatsheet

A comprehensive guide to navigating and utilizing command-line interfaces, covering essential commands, shell scripting, and environment management for increased productivity.

Navigation & File Management

Basic Navigation

`pwd`	Print Working Directory - Displays the current directory path.
`cd <directory>`	Change Directory - Navigates to the specified directory. Use `cd ..` to go up one level.
`ls`	List - Displays files and directories in the current directory. Use `ls -l` for detailed view, `ls -a` to show hidden files.
`.`	Represents the current directory.
`..`	Represents the parent directory.
`~`	Represents the user's home directory.

File Operations

`mkdir <directory>`	Make Directory - Creates a new directory.
`touch <file>`	Creates a new empty file.
`cp <source> <destination>`	Copy - Copies a file or directory from source to destination. Use `cp -r` for recursive copying of directories.
`mv <source> <destination>`	Move/Rename - Moves a file or directory, or renames it if the destination is in the same directory.
`rm <file>`	Remove - Deletes a file. Use with caution. Use `rm -r` to recursively delete directories, and `rm -rf` to force deletion without prompting.
`ln -s <target> <link_name>`	Create a symbolic link. A symbolic link (also known as a soft link) is a type of file that contains a reference to another file or directory in the form of an absolute or relative path.

File Content Examination

`cat <file>`	Concatenate - Displays the entire content of a file.
`head <file>`	Displays the first few lines of a file (default 10 lines). `head -n <number> <file>` displays the specified number of lines.
`tail <file>`	Displays the last few lines of a file (default 10 lines). `tail -n <number> <file>` displays the specified number of lines. `tail -f <file>` follows the file in real-time.
`less <file>`	Opens a file in a pager, allowing you to navigate through the content. Use `q` to quit.
`wc <file>`	Word Count - Displays the number of lines, words, and characters in a file.
`file <file>`	Determines the file type.

Searching & Filtering

Basic Searching

`grep <pattern> <file>`	Globally search a Regular Expression and Print. Searches for a specific pattern in a file. `grep -i` for case-insensitive search, `grep -r` for recursive search in directories.
`find <directory> -name <filename>`	Finds files in a directory hierarchy based on the specified name. `find . -type d` to find directories.
`locate <filename>`	Finds files by name using a pre-built database. Requires the `mlocate` package on many systems and database to be updated via `updatedb`.
`which <command>`	Locates the executable file associated with a command.
`whereis <command>`	Locates the binary, source, and manual page files for a command.
`history | grep <pattern>`	Searches command history for a specific pattern.

Filtering and Redirection

`|` (pipe)	Passes the output of one command as input to another command. Example: `ls -l | grep 'myfile'`
`>`	Redirects the output of a command to a file, overwriting the file if it exists. Example: `ls > filelist.txt`
`>>`	Appends the output of a command to a file. Example: `ls >> filelist.txt`
`2>`	Redirects standard error to a file. Example: `command 2> error.log`
`&>` or `>&`	Redirects both standard output and standard error to a file. Example: `command &> output.log`
`sort`	Sorts the lines of a text file. Example: `cat file.txt | sort`

Advanced Text Manipulation

`sed`	Stream EDitor - A powerful tool for text transformation. Example: `sed 's/old/new/g' file.txt` (replaces all occurrences of 'old' with 'new' in file.txt)
`awk`	Pattern scanning and processing language - Useful for extracting and manipulating data from text files. Example: `awk '{print $1}' file.txt` (prints the first column of each line)
`cut`	Removes sections from each line of files. `cut -d ',' -f 1,3 file.csv` (extracts the first and third fields from a comma-separated file)
`tr`	Translates or deletes characters. Example: `tr '[:lower:]' '[:upper:]' < file.txt` (converts all lowercase characters to uppercase)
`uniq`	Reports or omits repeated lines. Often used with sort. `sort file.txt | uniq`

Shell Scripting

Basic Script Structure

Every shell script typically starts with a shebang (`#!`) that specifies the interpreter to use.

```
#!/bin/bash
# This is a comment
echo "Hello, World!"
```

Make the script executable: `chmod +x <script_name>`

Variables

Variable Assignment	`variable_name="value"` (no spaces around `=`) Example: `name="John"`
Accessing Variables	`$variable_name` or `${variable_name}` Example: `echo "Hello, $name!"`
Read-only Variables	`readonly variable_name` Example: `readonly name`
Unsetting Variables	`unset variable_name` Example: `unset name`
Environment Variables	Variables that are set in the environment and available to all processes. Examples: `PATH`, `HOME`, `USER`

Control Structures

`if` statement

```
if [ condition ]; then
  # commands
elif [ condition ];
then
  # more commands
else
  # default commands
fi
```

`for` loop

```
for variable in list
do
  # commands
done
```

`while` loop

```
while [ condition ]
do
  # commands
done
```

`case` statement

```
case variable in
  pattern1)
    # commands
    ;;
  pattern2)
    # more commands
    ;;
  *)
    # default commands
    ;;
esac
```

Functions

Defining a Function

```
function_name() {
  # commands
}

#Or

function function_name {
  # commands
}
```

Calling a Function: `function_name`

Passing Arguments: Inside the function: `$1`, `$2`, etc.

Example: `function my_function { echo "First argument: $1" }`

Returning Values: Use `return value` (value must be an integer between 0 and 255). Use `echo` to return strings or other data.

Input/Output

Reading Input: `read variable_name`

Example:

```
echo -n "Enter your name: "
read name
echo "Hello, $name!"
```

Printing Output: `echo message`

Formatted output: `printf format arguments`

Example: `printf "Name: %s, Age: %d\n" "John" 30`

System Information & Process Management

System Information

`uname -a`	Displays kernel information.
`hostname`	Displays the system's hostname.
`uptime`	Shows how long the system has been running, along with the current time and average system load.
`df -h`	Displays disk space usage in a human-readable format.
`free -m`	Displays memory usage in megabytes.
`whoami`	Displays the current user.

Process Management

`ps`	Displays a snapshot of the current processes. `ps aux` for a more detailed view of all processes.
`top`	Displays a dynamic real-time view of running processes. Press `q` to quit.
`htop`	An interactive process viewer. Must be installed separately on most systems.
`kill <PID>`	Sends a signal to a process, usually to terminate it. Use `kill -9 <PID>` as a last resort.
`pkill <process_name>`	Kills processes by name.
`bg`	Resumes a suspended process in the background.
`fg`	Moves a background process to the foreground.
`jobs`	Lists the active jobs.

User and Group Management

`id`	Displays user and group IDs.
`groups`	Displays the groups a user belongs to.
`passwd`	Changes the user's password.
`sudo`	Executes a command with superuser privileges.
`su`	Substitute User - Allows switching to another user account.