Unix Shell Cheatsheet

A comprehensive cheat sheet for navigating and manipulating the Unix shell environment, covering essential commands, shortcuts, and scripting techniques.

Navigation & File Management

Basic Commands

`pwd`	Print working directory (shows the current directory).
`ls`	List directory contents (files and subdirectories). Options: `-l` (long listing), `-a` (all files, including hidden), `-t` (sort by modification time), `-h` (human-readable sizes).
`cd`	Change directory. `cd ..` (move up one level), `cd ~` (go to home directory), `cd -` (go to the previous directory).
`mkdir`	Create a new directory. `mkdir directory_name`
`rmdir`	Remove an empty directory. `rmdir directory_name`
`touch`	Create an empty file or update the timestamp of an existing file. `touch file_name`

File Operations

`cp`	Copy files or directories. `cp source_file destination_file`, `cp -r source_directory destination_directory` (recursive copy for directories).
`mv`	Move or rename files or directories. `mv source_file destination_file`, `mv old_name new_name`
`rm`	Remove files. `rm file_name`, `rm -r directory_name` (recursive removal for directories), `rm -f file_name` (force removal).
`cat`	Concatenate and display file contents. `cat file_name`
`head`	Display the beginning of a file. `head file_name` (first 10 lines), `head -n 20 file_name` (first 20 lines).
`tail`	Display the end of a file. `tail file_name` (last 10 lines), `tail -n 20 file_name` (last 20 lines), `tail -f file_name` (follow the file as it grows).
`less`	View file contents page by page. `less file_name`

Working with Text

Text Manipulation

`grep`	Search for patterns in files. `grep 'pattern' file_name`, `grep -i 'pattern' file_name` (case-insensitive), `grep -r 'pattern' directory_name` (recursive search).
`sed`	Stream editor for text manipulation. `sed 's/old/new/g' file_name` (replace all occurrences of 'old' with 'new').
`awk`	Pattern scanning and processing language. `awk '{print $1}' file_name` (print the first field of each line).
`wc`	Word count. `wc file_name` (lines, words, characters), `wc -l file_name` (lines only).
`sort`	Sort lines of text files. `sort file_name`, `sort -n file_name` (numeric sort), `sort -r file_name` (reverse sort).
`uniq`	Remove duplicate lines. `uniq file_name` (requires sorted input).
`cut`	Cut sections from each line of files. `cut -d ',' -f 1 file_name` (cut the first field using ',' as delimiter).

Redirection and Pipes

`>` - Redirect output to a file (overwrite).

Example:
`ls > file_list.txt`

`>>` - Redirect output to a file (append).

Example:
`ls >> file_list.txt`

`|` - Pipe the output of one command to another.

Example:
`ls -l | grep 'pattern'` (list files and filter the output).

`2>` - Redirect standard error to a file.

Example:
`command 2> error.log`

`&>` - Redirect both standard output and standard error to a file.

Example:
`command &> output.log`

System Information & Processes

System Info

`uname`	Print system information. `uname -a` (all information).
`df`	Display disk space usage. `df -h` (human-readable).
`du`	Estimate file space usage. `du -sh directory_name` (summary, human-readable).
`free`	Display amount of free and used memory. `free -m` (in MB), `free -g` (in GB).
`uptime`	Show how long the system has been running.
`whoami`	Print effective user ID.
`hostname`	Display the system's hostname.

Process Management

`ps`	Display running processes. `ps aux` (show all processes).
`top`	Display dynamic real-time view of running processes.
`kill`	Terminate a process. `kill PID` (sends TERM signal), `kill -9 PID` (sends KILL signal, forceful termination).
`jobs`	List active jobs.
`bg`	Put a job in the background. `bg %job_number`
`fg`	Bring a job to the foreground. `fg %job_number`
`nohup`	Run a command immune to hangups, with output to a non-tty. `nohup command &`

Shell Scripting

Basic Script Structure

```
#!/bin/bash

# Comments start with '#'

echo "Hello, world!"
```

Shebang (`#!/bin/bash`) indicates the interpreter for the script.

Variables:

```
NAME="John"
echo "My name is $NAME"
```

Command Substitution:

```
DATE=$(date)
echo "Today is $DATE"
```

Control Flow

`if` statement	`if [ condition ]; then`
	`  # code to execute if condition is true`
	`else`
	`  # code to execute if condition is false`
	`fi`

`for` loop	`for item in list;`
	`do`
	`  # code to execute for each item`
	`done`

`while` loop	`while [ condition ]; do`
	`  # code to execute while condition is true`
	`done`

`case` statement	`case variable in`
	`  pattern1)`
	`    # code to execute if variable matches pattern1`
	`    ;;`
	`  pattern2)`
	`    # code to execute if variable matches pattern2`
	`    ;;`
	`esac`

Functions

```
function_name() {
  # function body
  echo "Function called with arguments: $@"
  return 0
}

function_name arg1 arg2
```