Dash Shell Scripting Cheatsheet

A concise cheat sheet for Dash shell scripting, covering essential syntax, commands, and best practices for writing efficient and portable shell scripts.

Dash Fundamentals

Basic Syntax

Comments	`# This is a comment`
Variables	`variable=value` `echo $variable`
String literals	`'Single quoted'` `"Double quoted"`
Command Substitution	`$(command)` or `command`
Exit Status	`$?` (Exit status of last command)
Shebang	`#!/bin/sh` (Specifies the interpreter)

Control Structures

`if` statement	`if [ condition ]; then`
	`  # commands`
	`fi`

`if-else` statement	`if [ condition ]; then`
	`  # commands`
	`else`
	`  # commands`
	`fi`

`if-elif-else` statement	`if [ condition ]; then`
	`  # commands`
	`elif [ condition ]; then`
	`  # commands`
	`else`
	`  # commands`
	`fi`

`case` statement	`case $variable in`
	`  pattern1) commands ;;`
	`  pattern2) commands ;;`
	`  *) commands ;;`
	`esac`

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

`until` loop	`until [ condition ]; do`
	`  # commands`
	`done`

Commands and Utilities

File Manipulation

`ls`	List directory contents
`cp`	Copy files or directories
`mv`	Move or rename files or directories
`rm`	Remove files or directories
`mkdir`	Create directories
`touch`	Create an empty file or update timestamp

Text Processing

`echo`	Display a line of text
`cat`	Concatenate and display files
`grep`	Search for a pattern in files
`sed`	Stream editor for text manipulation
`awk`	Pattern scanning and processing language
`wc`	Word, line, and byte count

System Information

`uname`	Print system information
`date`	Print or set the system date and time
`pwd`	Print working directory
`whoami`	Print effective userid
`uptime`	Show how long the system has been running
`free`	Display amount of free and used memory

Expressions and Operators

Arithmetic Operators

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulo
`**`	Exponentiation (not POSIX standard, may not work on all shells)

String Operators

`=`	Equality
`!=`	Inequality
`-z`	True if string is empty
`-n`	True if string is not empty

File Test Operators

`-e file`	True if file exists
`-f file`	True if file exists and is a regular file
`-d file`	True if file exists and is a directory
`-r file`	True if file exists and is readable
`-w file`	True if file exists and is writable
`-x file`	True if file exists and is executable

Functions and Script Execution

Defining Functions

```
function function_name {
  # commands
}

# or

function_name () {
  # commands
}
```

Example:

```
my_function() {
  echo "Hello, world!"
}

my_function
```

Passing Arguments to Functions

Accessing arguments	`$1`, `$2`, … `$n`
All arguments	`$@`
Number of arguments	`$#`

Example

```
my_function() {
  echo "First argument: $1"
  echo "Second argument: $2"
}

my_function "arg1" "arg2"
```

Script Execution

Executing a script	`./script.sh`
	`# or`
	`sh script.sh`

Making a script executable	`chmod +x script.sh`

Sourcing a script	`. script.sh`
	`# or`
	`source script.sh`

Differences between executing and sourcing	Executing runs in a subshell, sourcing runs in the current shell.