BASH Cheat Sheet

BASH acronym for **The GNU Bourne-Again SHell** is an interface for you to execute statements, either at the interactive BASH prompt or via BASH scripts. It can run in interactive mode and also run in non-interactive mode when executing scripts. Scripts are lists of commands stored in a file. When you execute a script, all commands are executed one after another.

This Bash cheat sheet will show you all useful bash commands that any network or system admin can use as a quick reference.

Bash Scripting Basics

Here, we list some basic bash syntax with a brief explanation. It is a good starting point if you are a beginner.

Syntax	Explanation
#!/bin/bash	Used to tell the operating system the path it should use to interpret the file.
bash file.sh	Used to execute the script in the terminal.
./file.sh	Used to execute the script if it is executable.
#	Used to make comments in the script.
&&	logical AND operator.
| |	logical OR operator.
$#	Used to expands the number of arguments passed to the script.
$0	Used to expands to the name of the shell.
$1, $2	Used as an input parameter that you can add when running script.
exit [0-255]	Used to exit the script and return the number from 0 to 255.
$	Used for parameters and variables.

Syntax	Explanation
()	Used for running commands in a subshell.
$()	Used to save the output of commands.
(())	Used for arithmetic.
$(())	Used to retrieve the output of arithmetic expressions.
[]	Used in filename expansion and string manipulation.
<( )	It is very similar to a pipe and used for process substitution.
{ }	Used to expand sequences.
${ }	Used for string manipulation and variable interpolation.
|	Used to run multiple commands together.|
>	Used to send output to a file.
>>	Used to append output to a file.
;	Used to separate multiple commands.
<	Used to get input from a file.
~	Expands to the home directory.
~/.bashrc	Read by every non-login shell.
/etc/profile	Executed automatically at login.

File Test Operators

Here, we will list some helping testing operators for permissions, size, date, file type, or existence in the bash script.

Operators	Explanation
-e	To test if a file exists.
-f	To test if a given file is a regular file.

Operators	Explanation
-d	To test if the file is a directory.
-b	To test if the file is a block device.
-s	To test if the file is not zero sizes.
-L	To test if the file is a symbolic link.
-S	To test if the file is a socket.
-r	To test if the file has read permission.
-w	To test if the file has write permission.
-x	To test if the file has execute permission.
-g	Set group id on file or directory.
-u	Set user id on file or directory.
-k	Set a sticky bit.
-O	You are the owner of the file.
f1 -nt f2	file f1 is newer than f2.
f1 -ot f2	file f1 is older than f2

Comparison Operators

Comparison operators are used in bash to compare two strings to check if they are equal or not. Here, we will list some comparison operators including, string, and integer operators.

Integer Operators

Operators	Explanation
-eq	is equal to
-ne	is not equal to
-gt	is greater than

Integer Operators	
-ge	is greater than or equal to
-lt	is less than
-le	is less than or equal to

String Operators	
Operators	**Explanation**
=	is equal to
==	is equal to
!=	is not equal to
<	less than
<=	is less than or equal to
>	greater than
>=	is greater than or equal to
-z	string is null
-n	string is not null

Regular Expressions

Regular expressions are shortened as ‘regexp' or ‘regex'. They are strings of characters that define a search pattern. It can be used as a search or search & replace operation.

Expressions	Explanation
.	Matches any single character.
?	The preceding item is optional and will be matched, at most, once.
*****	The preceding item will be matched zero or more times.
+	The preceding item will be matched one or more times.

Expressions	Explanation
{N}	The preceding item is matched exactly N times.
{N,}	The preceding item is matched N or more times.
^	Matches the empty string at the beginning of a line.
$	Matches the empty string at the end of a line.
[a-d]	Matches any one character in the range a-d.

Loops and Conditions

A loop is a statement in a bash programming language that allows code to be repeatedly executed. You can set specific conditions during the script execution.

Loops	Explanation
if then fi	Used to test a condition.
if then else fi	Used to test a condition and use a fallback if the test fails.
if then elif else fi	Used to test a condition and use a fallback if all tests fail.
for do done	Iterate over a list of values.
while do done	Used to performs a given set of commands an unknown number of times as long as the given condition evaluates to true.
until do done	Used to execute a given set of commands as long as the given condition evaluates to false.
sleep time	Wait for a specified amount of time before continuing through the script.
break	Used to exit from the while or for loop but continue the rest of the script.
continue	Used to skip the current iteration of a loop and continue to the next iteration of the loop.

Bash Arrays and Functions

Array	Explanation
array=("elements of array")	Used to create an array of strings.
${array[0]}	Used to get the first element of the array.
${array[*]}	Used to get all values in the array.
${array[1]}	Get the last value in the array.
${array[@]}	Expand all of the array elements.
shift	Move argument from $2 to $1.
function() { content-of-function }	Used to define a function.
alias	Used to list all aliases defined in the current session.
alias alias='any command'	Used to define an alias.

Common Utilities and Switches

This cheat sheet will show you the most useful commands and switches to help you in your network and system administration.

Commands	Explanation
ls -l	List files by type and permission.
ls -a	List all files, including hidden files.
pwd	Display current working directory.
whoami	Who you are logged in as.
last	Display last user logins information.
find /home -name *.txt	Search all text files in /home directory.
find . -size 10k -print	Find all files greater than 10k in the current directory.
egrep "(foo|bar)" file.txt	Find the words foo and bar in file.txt.
sed s/foo/bar/g file.txt	Find the word foo and replace it with a bar in file.txt.

Commands	Explanation
locate file.txt	Find the location of the file.txt quickly.
grep foo file.txt	Searches the word foo in file.txt.
ps -ef	To check all running services.
netstat -ant	To check all network connections.
netstat -ent	To check established network connections.
ifconfig	To check all network interfaces, IPs, and Mac addresses.
ping	Used to check host reachability.
nslookup	Used for DNS query.
ssh	Used to login remote Linux system.
scp -r dir user@remote-ip:/opt/	Copy all files and directories recursively from the local system to a remote system.
scp -r user@remote-ip:/opt/ dir/	Copy all files and directories recursively from the remote system to a local system.
rsync -avz localdir user@remote-ip:/backup	Synchronize files/directories between the local and remote systems.
df -h	Shows free and used space on mounted filesystems.
du -sh	Shows total disk usage of the current directory.
free -m	Show free and used memory and swap space.
lsof	Lists files opened by running processes.
chown user:group filename	Change the owner of the file and directory.
chmod ugo file.txt	Change the user, group, and other permissions for file.txt.
kill pid	Kill any running process.
passwd username	Used to set or reset the user password.
top	Display all running processes, memory usage, cpu usage in real-time.

Shell Builtins

Builtin commands are contained within the shell itself. They called from a shell, that is executed directly in the shell instead of an external executable program.

Builtins	Explanation
.	Used to reads and runs commands from a designated file in the current shell.
alias	Used to define an alias for a specific command.
bg	Run a job in background mode.
bind	Used to bind a keyboard sequence.
break	Used to exit from a running loop in script.
cd	Change the directory to another directory.
command	Run a specific command without the normal shell lookup.
continue	Resumes the next iteration of the loop in script.
declare	Used to declare a variable.
dirs	Shows a list of all remembered directories.
disown	Remove a job from the job table.
enable	Used to enable or disable built-in command.
exec	Replace the shell process with the specified command.
exit	Used to exit the shell with an exit status.
export	Used to set a variable available for sub-process.
fc	Select a list of commands from the history list.
fg	Run a job in foreground mode.
hash	Used to find and remember the full path of the specified command.
help	Used to display the help file.
history	List the history of all commands.

Builtins	Explanation
jobs	List all active jobs.
logout	Used to exit from the current shell.
pwd	Display the path of the current working directory.
read	Read one line from STDIN and assigns it to a variable.
popd	Removes entries from the directory stack.
pushd	Add a directory to the directory stack.
printf	Displays text in a formatted string.
source	Read and executes commands from a specified file in the current shell.
times	Displays the accumulated user and system shell time.
wait	Make the shell wait for a job to finish.