

Scripting Utilities Cheatsheet

A comprehensive cheat sheet covering various scripting utilities, including `xargs`, `find`, `sed`, `awk`, `grep`, and `jq`. This cheat sheet provides a quick reference to essential commands, options, and examples to help you automate tasks and manipulate data efficiently.



xargs & find

xargs Basics

xargs Build and execute command lines from standard input. Takes input from stdin and converts it to arguments for a command. General syntax. If command is omitted, xarqs [options xargs defaults to /bin/echo.] [command 1 Use at most max-args arguments per -n maxargs command line. Replace occurrences of replace-str in - I replacethe initial-arguments with names read from str standard input. Also implies -x and -L 1. Use at most max-lines non-blank input -L maxlines lines per command line. -d Input items are terminated by the specified character. Useful when filenames contain delimite spaces. r

find Basics

find [path] [expressio n]	Search for files in a directory hierarchy.
-name	Base of file name (the path with the leading directories removed) matches shell pattern pattern.
-type	File is of type type: f: regular file d: directory 1: symbolic link
-mtime n	File's data was last modified n*24 hours ago.
-exec command {}	Execute command ; all matched files will be appended to the end of the command.
-delete	Delete files; be careful when using this option.

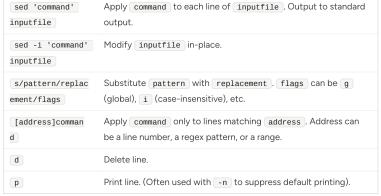
Combining xargs and find

xargs to process them.		
findname "*.txt" -print0 xargs -0 wc -l		
This command finds all .txt files in the current		
directory and its subdirectories, and then counts the		
number of lines in each file using wc -1print0 and		
number of lines in each file using wc -1printo and		
-0 handle filenames with spaces correctly.		

Common use case: using find to locate files and

sed

sed Basics



sed Examples

```
sed 's/foo/bar/g' file.txt

Replace all occurrences of foo with bar in file.txt and print to standard output.

sed -i 's/foo/bar/g' file.txt

Replace all occurrences of foo with bar in file.txt in-place.

sed '/^#/d' file.txt

Delete all lines starting with #.

sed -n '/pattern/p' file.txt

Print only lines that match pattern.

sed '2,5d' file.txt

Delete lines 2 through 5.

sed '$d' file.txt

Delete the last line.
```

awk

Page 1 of 3 https://cheatsheetshero.com

awk Basics awk Examples

<pre>awk 'pattern { action }' file</pre>	Process file line by line. If pattern matches, execute action.
BEGIN { action }	Execute action before processing any lines.
END { action }	Execute action after processing all lines.
\$0	The entire line.
\$1, \$2,	The first, second, etc. field (column) in the line.
NF	Number of fields in the current line.

```
awk '{ print $1 }' file.txt

Print the first field of each line.

awk '{ print $NF }' file.txt

Print the last field of each line.

awk '/pattern/ { print }' file.txt

Print all lines that match pattern.

awk '$1 > 10 { print }' file.txt

Print all lines where the first field is greater than 10.

awk 'BEGIN { sum = 0 } { sum += $1 } END { print sum }' file.txt

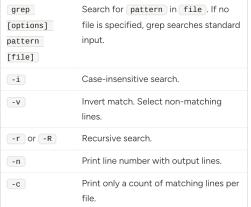
Calculate the sum of the first field of all lines.

awk -F',' '{ print $2 }' file.csv

Print the second field of each line in a CSV file, using , as the field separator.
```

grep & jq

grep Basics



grep Examples

grep 'foo' file.txt Print all lines in file.txt that contain foo. grep -i 'foo' file.txt Print all lines in file.txt that contain foo, case-insensitive. grep -v 'foo' file.txt Print all lines in file.txt that do not contain foo. grep -r 'foo'.

```
grep -r 'foo' .

Recursively search for foo in all files in the current directory.

grep -n 'foo' file.txt

Print all lines in file.txt that contain foo , along with their line numbers.
```

Print the number of lines in file.txt that contain

grep -c 'foo' file.txt

foo .

jq Basics

<pre>jq [options] 'filter' [file]</pre>	JSON processor. If no file specified, reads from stdin.
	The identity filter. Outputs the input as is.
. key	Access the value associated with the key key.
[].	Access all elements in an array.
	Pipe filters.
raw-output Or	Output raw strings, not JSON.

Page 2 of 3

jq Examples

```
pretty-print the JSON in data.json.

jq '.name' data.json

Extract the value associated with the key name.

jq '.users[]' data.json

Extract all elements from the users array.

jq '.users[].name' data.json

Extract the name field from each element in the users array.

jq '[.[] | .age]' data.json

Extract the age from each element of the top-level array.

curl -s https://api.example.com/data | jq '.[] | .title'

Fetch data from an API and extract the title field from each element in the resulting array.
```