

### Install zawk

```
$ cargo install zawk
$ brew install linux-china/tap/zawk
$ sudo xattr -r -d com.apple.quarantine $(readlink -f $(brew --prefix zawk))/bin/zawk
```

### how to run zawk?

```
$ zawk 'BEGIN { print strftime(), whoami() }'
$ zawk -f demo.awk demo.txt
```

### String functions: utf-8 by default

length(s)	Length of s: utf-8 format
char_at(\$1, 1)	Get char at index, starts from 1
match(s, re)	if string s matches the regular expression
substr(s, i[, j])	1-indexed substring of string s
sub(re, t, s)	Substitutes t for the first matching occurrence of regular
gsub(re, t, s)	Like sub, but with all occurrences substituted, not just the first.
index(s,t)	Position in string s where string t occurs, 0 if not found
last_index(s,t)	Last position in string s where string t occurs, 0 if not found
split(s, m[, fs])	Splits the string s according to fs, placing the results in the array m.
sprintf(fmt, s, ...)	Returns a string formatted according to fmt and provided arguments
printf(fmt, s, ...) [>[>] out]	Like sprintf but the result of the operation is written to standard output
hex(s)	Returns the hexadecimal integer (e.g. 0x123abc)

### String functions: utf-8 by default (cont)

join_fields(i, j[, sep])	Returns columns i through j (1-indexed, inclusive) concatenated
join_csv(i, j)	Like join_fields but with columns joined by comma
join_tsv(i, j)	Like join_fields but with columns joined by tabs
tolower(s)	Returns a copy of s where all lowercase ASCII characters
toupper(s)	Returns a copy of s where all uppercase ASCII characters
strtonum(s)	numeric value(Decimal) strtonum("0x11")
trim(s), trim(s, "[ ]")	Trim text with space by default
truncate(s,10)	Truncate s with width fixed
capitalize(s)	Capitalize first character of s
uncapitalize(s)	Uncapitalize first character of s
camel_case(s)	Return camel case of s: helloWorld
kebab_case(s)	Return kebab case of s: hello-world
snake_case(s)	Return snake case of s: hello_world
title_case(s)	Return kebab case of s: Hello World
isint(s)	Integer(64) or not for s
isnum(s)	Number(int or float) or not for s
starts_with(s, prefix)	Check s start with prefix or not
ends_with(s, suffix)	Check s end with prefix or not
contains(s, child)	Check s contains with prefix or not



### String functions: utf-8 by default (cont)

mask(s)	Mask some sensitive text, such as email, phone
pad(s, 10, "**")	Pad s with width and place holder
strcmp(s1, s2)	Compare 2 text
lines(text)	Split text into none-empty lines
words(s)	Text to words array
repeat("**", 3)	Repeat text with times
default_if_empty(s, "0")	Return default value if text is empty or not exist.
append_if_missing(s, "/")	Add suffix if missing
prepend_if_missing(s, "https://")	Add prefix if missing
remove_if_begin("./demo.json", "./")	Remove prefix if available
remove_if_end("demo.json", ".json")	Remove suffix if available
quote(s)	Quote text if not quoted.
double_quote(s)	Double quote text if not quoted.
format_bytes(size)	Format bytes size: 10.1 MB
to_bytes("10.2 MB")	Convert bytes format to size
escape(format, s)	Escape s with format: json, csv, tsv, xml, sql, shell
escape_csv(s)	Escape s with CSV
last_part(text, sep)	Get last part with seperator
parse(text, template)	Parse text with wild match
rparse(text, regex)	Parse text with regex match group
fake(data, locale)	Generate fake data by locale
mkpass(length)	Generate password
figlet(text)	Generate figlet ascii art

- parse: use wild match - `parse( " Hello World", " {greet} {name} ") ["g ree t"]`

- rparse: use regex group - `rparse ("Hello World", " (\\w+ ) (\\w+) ") [1]`

# fake function restriction

\* data: name, phone, cell, email, ip,creditcard,wechat,id,zipcode,plate

\* locale: en, cn

### Text Parser

url("https://example.com/path")	Parse s to URL array: schema, user, host, port ...
data_url("data:text/plain;base64,SGVsbG8sIFdvcmxkIQ==")	Parse data url: data_url("data:text/plain;base64,SGVsbG8sIFdvcmxkIQ==")
shlex("ls -al")	Parse command line to array
path("./demo.txt")	Parse path
semver("1.2.3-alpha")	Parse semantic version
pairs("id=1&name=Hello%20Wo")	Parse url query or normal pairs: "a=b,c=d"
record("requests_total{code=\"200\"}")	Parse pairs with name, such as Prometheus
message("name{a=1} (body)")	Parse message: name, headers and body (body)"
func("hello(1,2,3)")	Parse function call
flags("{vip,top20}")	Parse flags
variant("week(5)")	Parse variant
tuple(("first", 1))	Parse tuple
parse_array("[first', 'second']")	Parse array

### I/O functions

read_all(file_path)	Read file into text
write_all(file_path, text)	Write text into file
getline()	Read line from file
read_config(file_path)	Read ini, properties into Map

getline:

- [https://www.gnu.org/software/gawk/manual/html\\_node/Getline.html](https://www.gnu.org/software/gawk/manual/html_node/Getline.html)  
 - <http://awk.freeshell.org/AllAboutGetline>

### OS functions

system(cmd)	Execute cmd and return exit status
whoami()	user name
os()	OS name
arch()	such as x86_64, aarch64 ...
os_family()	unix or windows
pwd()	Current working directory
user_home()	User home directory



### Database functions

<code>sqlite_query("sqlite.db",sql)</code>	SQLite query: return value is array of CSV lines
<code>sqlite_execute("sqlite.db", sql)</code>	SQLite execute
<code>libsql_query("http://localhost:8080",sql)</code>	libSQL query: return value is array of CSV lines
<code>libsql_execute("http://localhost:8080", sql)</code>	libSQL execute
<code>mysql_query(url, sql)</code>	MySQL query: return value is array of CSV lines
<code>mysql_execute(url, sql)</code>	MySQL execute
<code>pg_query(url, sql)</code>	PostgreSQL query: return value is array of CSV lines
<code>pg_execute(url, sql)</code>	PostgreSQL execute
MySQL url: <code>mysql://root:123456@localhost:3306/test</code>	
PostgreSQL url: <code>postgres://postgres:postgres@localhost/db_name</code>	

### Misc commands

```
# dump prometheus text to CSV
$ zawk dump --prometheus http://localhost:8080/actuator/prometheus
# parse CSV
$ zawk -f demo.awk -i csv demo.csv
# Nushell
$ ls | to csv | ^zawk -i csv '{print $1}'
```

### AWK & Friends

grep	<a href="https://www.gnu.org/s/grep/manual/grep.html">https://www.gnu.org/s/grep/manual/grep.html</a>
sed	<a href="https://www.gnu.org/software/sed/manual/sed.html">https://www.gnu.org/software/sed/manual/sed.html</a>
DuckDB	<a href="https://duckdb.org/">https://duckdb.org/</a>
xsv	<a href="https://github.com/BurntSushi/xsv">https://github.com/BurntSushi/xsv</a>
jq	<a href="https://jqlang.github.io/jq/">https://jqlang.github.io/jq/</a>

### ID generator

<code>uuid()/uuid("v7")</code>	Generate uuid: 128 bits
<code>ulid()</code>	Generat ulid: 128 bits
<code>tsid()</code>	Generate TSID
<code>snowflake(machine_id)</code>	Generate snowflake: 64 bits, max value for machine_id is 65535

### Array functions

<code>length(arr)</code>	Length of array
<code>delete arr[1] / delete arr</code>	Delete array item or array
<code>seq(start, end, step)</code>	Generate sequence array: seq command compatible
<code>uniq(arr)</code>	Unique items of array: uniq command compatible
<code>_ = asort(arr)</code>	Sort array items by asc
<code>_max(arr)</code>	Return max value of number array
<code>_min(arr)</code>	Return min value of number array
<code>_sum(arr)</code>	Return sum value of number array
<code>_mean(arr)</code>	Return man value of number array
<code>_join(arr, ",")</code>	Join array items to string
<code>bf_insert(item)</code>	Bloom filter insert
<code>bf_contains(item)</code>	Bloom filter contains
<code>bf_icontains(item)</code>	Bloom filter contains with insertion if not found

Bloom filter:

- `bf_insert(item,group)`
- `bf_contains(item,group)`
- `bf_icontains(item,group)`

### Math functions

<code>rand()</code>	Random float number between 0 and 1
<code>srand(x)</code>	Seeds the random number generator used by rand
<code>abs(x)</code>	Absolute value of x
<code>floor(x)</code>	Return int floor value of x
<code>ceil(x)</code>	Return int ceil value of x
<code>round(x)</code>	Return int round value of x
<code>fend(expression)</code>	Calculate by fend: <a href="https://github.com/printfn/fend">https://github.com/printfn/fend</a>
<code>min(x,y,z)</code>	Return min value of x, y, z
<code>max(x,y,z)</code>	Return max value of x, y, z
<code>mkbool(s)</code>	Return 0 or 1 for bool string: false, true, Y, N ...
<code>int(s)</code>	Convert s to int
<code>float(s)</code>	Convert s to float

### Date/Time functions

systeme()	Current unix time
strftime()/strftime(p, timestamp)	Format timestamp by pattern
mktime(s)	Parse s to unix timestamp
datetime(s)	Parse s to date/time array
duration("2min + 12sec")	Parse expression to duration in seconds

strftime pattern: <https://docs.rs/chrono/latest/chrono/format/strftime/index.html>

date/time parse: <https://docs.rs/dateparser/latest/dateparser/#accepted-date-formats>

date/time array:

- year: 2024
- month: 1, 2
- monthday: 24
- hour
- minute
- second
- yearday
- weekday
- hour: 1-24
- althour: 1-12

### JSON/CSV/XML/HTML functions

from_json(json_text)	Parse json text to array
to_json(arr)	Output array as json text
json_value(json_text, json_path)	Get one text value by json path
json_query(json_text, json_path)	Query text values by json path
from_csv(line)	Parse CSV line to array
to_csv(arr)	Output array as CSV line
xml_value(xml_text, xpath)	Get text value by xpath
xml_query(xml_text, xpath)	Query text values by xpath
html_value(html_text, selector)	Get text value by css selector
html_query(html_text, selector)	Query text values by css selector

### Encode/Decode functions

encode(format, s)	Encode s with format
decode(format, s)	Decode s with format

Formats:

- hex, url, base32/58/64, base64url
- base64-hex, hex-base64
- zlib2base64url: zlib then base64url, good for online diagram service, such as PlantUML, Kroki

### Crypto functions

digest(algorithm, s)	Digest s with algorithm
hmac(algorithm, secret-key, s)	hmac s with: HmacSHA256, HmacSHA512
jwt("HS256", secret-key, arr-payload)	Generate JWT token: HS256, HS384, HS512
dejwt(secret-key, token)	Verify JWT token and return payload array
encrypt("aes-128-cbc", "Secret Text", "pass_key")	Encrypt secret text
decrypt("aes-128-cbc", "7b9c07..", "pass_key")	Decrypt to plain text

Digest algorithm:

- md5, sha256, sha512
- bcrypt
- murmur3, xxh32, xxh64, blake3
- crc32, adler32

AES algorithm:

- aes-128-cbc, aes-256-cbc
- aes-128-gcm, aes-256-gcm

iv required gcm: encrypt("aes-128-gcm", "Secret Text", "pass\_key", "your\_iv")

JWT algorithm

- HS256, HS384, HS512
- RS256, RS384, RS512, ES256, ES384, EdDSA: please private/public key PEM text as key
- JWK: for dejwt only, and key is alike <https://xxx.comple.com/j-wks.json#kid>

### Network functions

http_get(url, headers)	Return HTTP response
http_post(url, headers, body)	Return HTTP response
send_mail(from, to, subject, body)	Send email by REST
smtp_send(url, from, to, subject, body)	Send email by SMTP
s3_get(bucket, object_name)	Return the text value of object
s3_put(bucket, object_name, body)	Put the text body to s3 bucket
publish(nats_mqtt_url, body)	Publish message to NATS

Environment variables for "send\_mail":

- MLSN\_A PI\_KEY: API key for MailerSend

url for "smtp\_send":

- SMTP basic: smtp://localhost:1025

```
- SMTP + TLS smtps: //u ser :pa ssw ord @ho st:465
```

Environment variables for S3 access:

- S3\_ENDPOINT
- S3\_ACCESS\_KEY\_ID
- S3\_ACCESS\_KEY\_SECRET
- S3\_REGION

url for "publish":

- NATS: nats://host:4222/topic
- MQTT: mqtt://servername:1883/topic
- MQTTS: mqtts://token@YOUR-BROKER.YOUR-NAMESPACE.c-loudflarepubsub.com/topic



By **linux\_china**

[cheatography.com/linux-china/](https://cheatography.com/linux-china/)

Published 15th March, 2024.

Last updated 7th October, 2024.

Page 4 of 6.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### KV functions

<code>kv_get(ns, key)</code>	Get value by namespace and key
<code>kv_put(ns, key, value)</code>	Put value by namespace and key
<code>kv_delete(ns, key)</code>	Delete value by namespace and key
<code>kv_clear(ns)</code>	Clear all keys

KV support by SQLite, Redis and NATS:

- SQLite: ns is normal name, such as "cluster1", "app1"
- Redis: ns is redis://localhost:6379/0/namespace
- NATS: ns is nats://localhost:4222/bucket\_name

### Color

<code>hex2rgb("#FF0000")</code>	Convert hex color to RGB array
<code>rgb2hex(r,g,b)</code>	Convert RGB to hex

### Misc functions

<code>var_dump(value)</code>	Dump and output variable to console with json format
<code>log_debug(msg)/log_info/log_warn/log_error</code>	Log s and output to console
<code>isarray(x)</code>	Is array or not
<code>typeof(x)</code>	Type name of x: array, number, string, unassigned

### References

AWK Cheat Sheet	<a href="https://cheatsheets.zip/awk">https://cheatsheets.zip/awk</a>
Awk in 20 Minutes	<a href="https://ferd.ca/awk-in-20-minutes.html">https://ferd.ca/awk-in-20-minutes.html</a>
Gawk: Effective AWK Programming	<a href="https://www.gnu.org/software/gawk/manual/">https://www.gnu.org/software/gawk/manual/</a>



By [linux\\_china](#)  
[cheatography.com/linux-china/](https://cheatography.com/linux-china/)

Published 15th March, 2024.  
Last updated 7th October, 2024.  
Page 5 of 6.

Sponsored by [CrosswordCheats.com](#)  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>