Cheatography

zawk Cheat Sheet

by linux_china via cheatography.com/141366/cs/42735/

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
chat_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

By **linux_china**
cheatography.com/linux-china/

Published 15th March, 2024.
Last updated 7th October, 2024.

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
preappend_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,xxxx")	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
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
- 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

zawk Cheat Sheet

by linux_china via cheatography.com/141366/cs/42735/

Database functions

sqlite_query("sqlite.db",sql)	SQLite query: return value is array of CSV lines
sqlite_execute("sqlite.db", sql)	SQLite execute
libsql_query("http://localhost:8080",sql)	libSQL query: return value is array of CSV lines
libsql_execute("http://localhost:8080", sql)	libSQL execute
mysql_query(url, sql)	MySQL query: return value is array of CSV lines
mysql_execute(url, sql)	MySQL execute
pg_query(url, sql)	PostgreSQL query: return value is array of CSV lines
pg_execute(url, sql)	PostgreSQL execute

MySQL url: mysql://root:123456@localhost:3306/test
PostgreSQL url: postgresql://postgres:postgres@localhost/db_name

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	https://www.gnu.org/s/grep/manual/grep.html
sed	https://www.gnu.org/software/sed/manual/sed.html
DuckDB	https://duckdb.org/
xsv	https://github.com/BurntSushi/xsv
jq	https://jqlang.github.io/jq/

ID generator

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

Array functions

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

Bloom filter:
- bf_insert(item,group)
- bf_contains(item,group)
- bf_icontains(item,group)

Math functions

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

Cheatography

zawk Cheat Sheet

by linux_china via cheatography.com/141366/cs/42735/

Date/Time functions

systime()	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

Cryto 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/jwks.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 "smpt_send":
- SMTP basic: `smtp:/ /lo cal hos t: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

Cheatography

zawk Cheat Sheet

by linux_china via cheatography.com/141366/cs/42735/

KV functions

kv_get(ns, key)	Get value by namespace and key
kv_put(ns, key, value)	Put value by namespace and key
kv_delete(ns, key)	Delete value by namespace and key
kv_clear(ns)	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

hex2rgb("#FF0000")	Conert hex color to RGB array
rgb2hex(r,g,b)	Convert RGB to hex

Misc functions

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

References

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