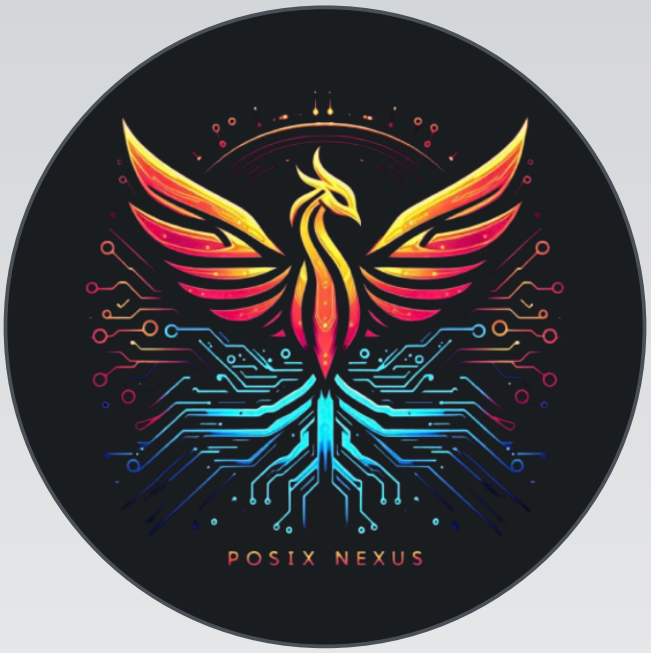
Posix-Nexus Shell



Canine-Table

March 27, 2025

Contents

I Str

I Str

The following functions enable dynamic string manipulation, including random string generation, targeted searches and replacements, and case adjustments for versatile text handling.

- **get_str_rand()**: Generates a random string of specified length **num**, using character sets **chars** such as alphanumeric or others.
- **get_str_locate()**: Searches for occurrences of a string **fnd** within the input, optionally replacing it with **rpl**, separating content with **sep**, and supporting global or targeted searches.
- **set_str_case()**: Modifies the case of a string, converting it to uppercase, lowercase, or title case, based on the provided option (`u`, `l`, or `t`).
- **set_str_format()**: Formats a string based on a specified format **fmt**, with optional separators **sep**, and alignment options like **lft**, **rgt**, or **kp**.
- **add_str_append()**: Appends a specified character **char** or string multiple times to reach a desired length **num**, optionally extending or modifying the input based on **ed**.
- **add_str_div()**: Creates a horizontal divider string of the length derived from terminal column size, using repeated characters like `"-"`.

II Algorithms

II Algorithms

] The following functions utilize efficient algorithms for sorting and processing data structures, with a focus on modularity and adaptability.

- **set_algor_qsort()**: Implements a QuickSort algorithm to sort a list (**lst**) of elements, with options to reverse the sort order (**rvs**), apply a custom sorting mechanism (**meh**), and use specified delimiters (**sep** and **osep**).

III Int

III Int

The following functions provide powerful computational tools for performing advanced numerical operations, including base-specific arithmetic, distribution, range adjustment, and mathematical constants handling.

- **get_int_conv()**: Converts a number **num** from its original base **from** to another base **to**, supporting optional signed number handling.
- **get_int_bsubt()**: Computes the difference of two numbers, **minuend** and **subtrahend**, in base **from** with a specified precision **prec**, supporting signed numbers.
- **get_int_badd()**: Computes the sum of two numbers, **addend1** and **addend2**, in base **from** with a specified precision **prec**, supporting signed numbers.
- **get_int_comp()**: Computes the complement of a number **num** in the specified **base**, leveraging AWK utility functions for base-specific computations.
- **get_int_abs()**: Calculates the absolute value of **num**, ensuring the result is always a positive number, using AWK's utility functions.
- **get_int_fact()**: Computes the factorial of **num**, with an option to print intermediate steps if **prnt** is set to true.
- **get_int_fib()**: Computes the **num**-th Fibonacci number, optionally printing intermediate sums if **prnt** is set to true.
- **get_int_round()**: Rounds **num** according to the specified method **rnd** (e.g., `ceiling` or `round`), defaulting to truncation if no method is provided.
- **get_int_gcd()**: Computes the greatest common divisor (GCD) of two numbers, **num1** and **num2**, using the Euclidean algorithm.
- **get_int_remainder()**: Computes the remainder of dividing **num1** by **num2**, ensuring both inputs are valid digits.
- **get_int_lcd()**: Calculates the least common denominator (LCD) of **num1** and **num2** using AWK's mathematical utilities.
- **get_int_tau()**: Returns the value of τ (the circle constant, $\tau = 2\pi$), optionally based on the input **num** for calculations or prints a default τ if no input is provided.
- **get_int_pi()**: Returns the value of π (pi constant), optionally using the input **num** for calculations or defaults to a general π value when no input is specified.

III Int

- **get_int_distribute()**: Distributes **num1** evenly across the range defined by **num2** and **num3**, ensuring all inputs are valid digits.
- **get_int_range()**: Adjusts **num1** to fit within the range defined by **num2** and **num3**, using modulus operations for precise computation.

IV Struct

IV Struct

The following functions provide a robust set of tools for managing structured data in shell scripts, covering retrieval, comparison, manipulation, and execution, with a focus on modularity and efficiency.

- **get_struct_ref()**: Retrieves the value of a variable by its name, allowing for dynamic access and reference in shell scripts.
- **get_struct_ref_append()**: Appends a value to the referenced variable, optionally inserting a separator before the new content, and returns the updated structure.
- **get_struct_compare()**: Compares two structures (input list and reference list), with options for case sensitivity, delimiters, and comparison modes (e.g., left, right, or intersection).
- **get_struct_list()**: Processes an input list with options for reversing, deduplication, or restructuring, while using specified separators for splitting and joining elements.
- **new_struct_task()**: Executes tasks iteratively on elements from a structured list, with configurable input, output, and error streams, as well as background execution control.
- **set_struct_noexpand()**: Prepares a variable for structured assignment by escaping special characters, ensuring its value is preserved in a non-expanded format.
- **set_struct_opt()**: Processes input and reference lists (**inpt** and **reflst**) using specified delimiters and options, matching input against reference values with configurable verbosity, case sensitivity, and length validation.

^ IV Struct

- **get_int_range()**: Adjusts **num1** to fit within the range defined by **num2** and **num3**, using modulus operations for precise computation.

V Content

V Content

] The following functions utilize efficient algorithms for sorting and processing data structures, with a focus on modularity and adaptability.

- **set_algor_qsort()**: Implements a QuickSort algorithm to sort a list (**lst**) of elements, with options to reverse the sort order (**rvs**), apply a custom sorting mechanism (**meh**), and use specified delimiters (**sep** and **osep**).

VI Cmd

VI Cmd

The following functions offer essential utilities for discovering and verifying the availability of commands, enhancing the portability and adaptability of shell scripts across different environments.

- **get_cmd()**: Iterates through a list of commands provided as arguments, checks their availability using `command -v`, and returns the first found command or exits if none are found.
- **get_cmd_pager()**: Searches for commonly used pager commands (`less`, `more`, and `tee`) by leveraging the **get_cmd()** function.
- **get_cmd_awk()**: Searches for AWK implementations (`mawk`, `nawk`, `awk`, `gawk`) using the **get_cmd()** function.
- **get_cmd_shell()**: Searches for available shell interpreters (`dash`, `sh`, `bash`, `zsh`, `fish`, and others) in the current environment.
- **get_cmd_editor()**: Locates command-line text editors (`nvim`, `vim`, `gvim`, `vi`) for editing files.
- **get_cmd_tex_compiler()**: Searches for LaTeX compilation utilities (`latexmk`, `pdflatex`, `lualatex`, `xelatex`).
- **get_cmd_pdf_viewer()**: Finds installed PDF viewers (`zathura`, `mupdf`, `evince`).
- **get_cmd_pkgmgr()**: Searches for package management tools (`pacman`, `apt`, `dnf`, `brew`, and others) in the system.

VII Tty

VII Tty

The following functions provide utilities for managing and querying TTY properties, including property retrieval, structured formatting, and signal handling for enhanced user interaction.

- **get_tty_prop_list()**: Lists all TTY properties in key-value pairs, processing the output of `stty -a` for structured formatting.
- **get_tty_prop()**: Retrieves specific TTY properties based on provided keys (`-k`) or values (`-v`), enabling focused property queries.
- **set_tty_hault()**: Temporarily disables the cursor using `setterm` and traps signals to re-enable it upon script exit or interruption.

VIII Pkgmgr

VIII Pkgmgr

] The following functions serve as wrappers for various package managers, offering a unified interface for common operations like updating, searching, installing, and managing software packages across different environments.

- **get_pkgmgr()**: A wrapper function for interacting with the defined package manager, supporting operations like updating (`-u`), querying (`-q`), searching (`-s`), installing (`-i`), removing (`-r`), and cleaning caches (`-c`).
- **__set_pkgmgr()**: Manages the execution of package manager commands by mapping user-specified options to the corresponding commands for the chosen package manager.
- **__get_pkgmgr_*()**: Defines package manager-specific command mappings for each supported package manager, such as `pacman`, `apt`, `apk`, `brew`, and others.