
Hako User Manual

Niall Ryan (21454746), Cathal O’Grady



2025-04-28

Contents

Hako User Manual	2
Hako	2
Target Readers	2
Content	2
Building	2
Dependencies	2
Build instructions	3
Running	3
Development Server (Easiest)	3
Self-Hosting	4
Features	4
Desktop	4
Applications	8
Core-Utils	16
System APIs	16

Hako User Manual

Hako

Hako is a platform agnostic Unix-like integrated development platform for the purposes of educating the youth about programming in a more “systems-oriented” way.

Target Readers

This manual is intended for Hako’s two target audiences;

- Students
- Mentors
- Contributors

Content

If you want to build from source, and host locally, this manual will describe:

- How to build hako from source
- How to run it locally

This manual will also describe the features of Hako;

- Applications
- Core-Utils
- System API

Building

Dependencies

Build instructions applicable Unix/Posix/Linux environments

- Justfile
- Ldoc (lua-Ldoc)
- Meson + Ninja (fortnite)
- Gcc
- Emscripten

- Node (npm)
- gpg
- cp

Build instructions

```
1 just
```

You can do a clean build with:

```
1 just clean && just
```

You can also build the site as a static bundle with:

```
1 just site
```

You can then run it with any webserver. Just make sure that the following response headers are added:

```
1 Cross-Origin-Embedder-Policy require-corp
2 Cross-Origin-Opener-Policy same-origin
```

Running

Development Server (Easiest)

If you wish to simply run Hako for personal use, you can run it using a development server.

A development server is a simple server on your own machine for personal use, and allows you to use the application easily - often used for the actual development of Hako.

Once you build Hako (see build instructions above), a development server can be ran via:

```
1 just site-run-dev
```

or more succinctly

```
1 just srd
```

It will then be available on 127.0.0.1:5173 (localhost:5173) in your browser.

Self-Hosting

Hako is incredibly simple to self-host as it is **completely client-side**, which means it all runs on your device. There are no servers or any other compute required to run the application.

If you build the site (see build instructions above), you have a fully self-contained static bundle of Hako. It contains all the HTML, JS, CSS, Web Assembly, etc required for Hako to run.

This can be exposed via a simple web server (like Nginx, Apache, Vercel, etc).

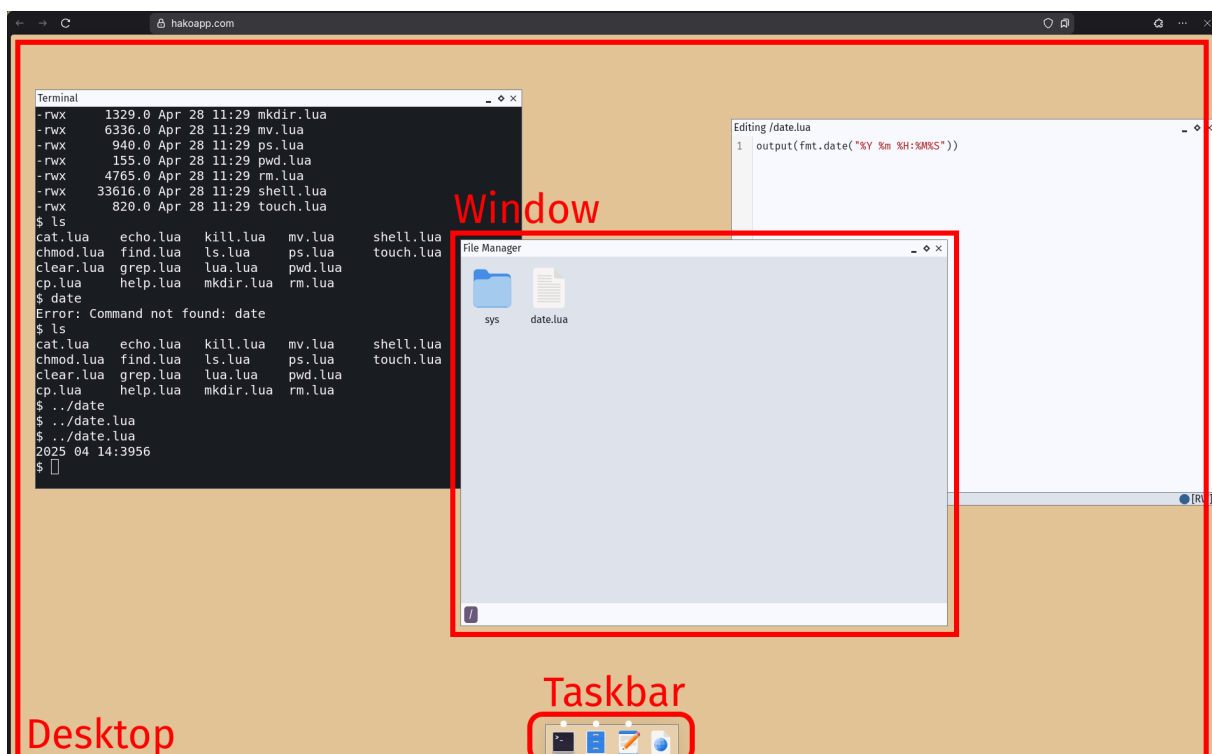
Features

Hako has a whole bunch of features to assist you in learning (or teaching) systems programming!

A high-level category of these features are:

- **The Desktop** – The graphical user interface for the system
- **Applications** – The applications available for you to use
- **Core-Utils** – Useful tools and utilities for you to use in the terminal
- **System APIs** – Interfaces to manipulate and use your system with

Desktop

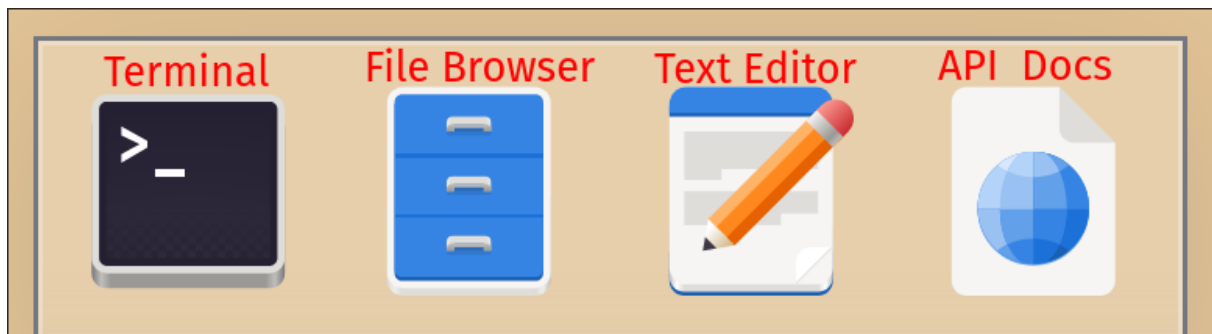


The desktop should be a familiar interface to those who have used a typical operating system before (such as Windows, MacOS, ChromeOS).

It is a “floating window manager”, which means windows can:

- Be dragged and placed anywhere on the screen
- Overlap eachother (windows can be on top of others)
- Be manually resized and positioned
- No strict rules are automatically enforced

Taskbar



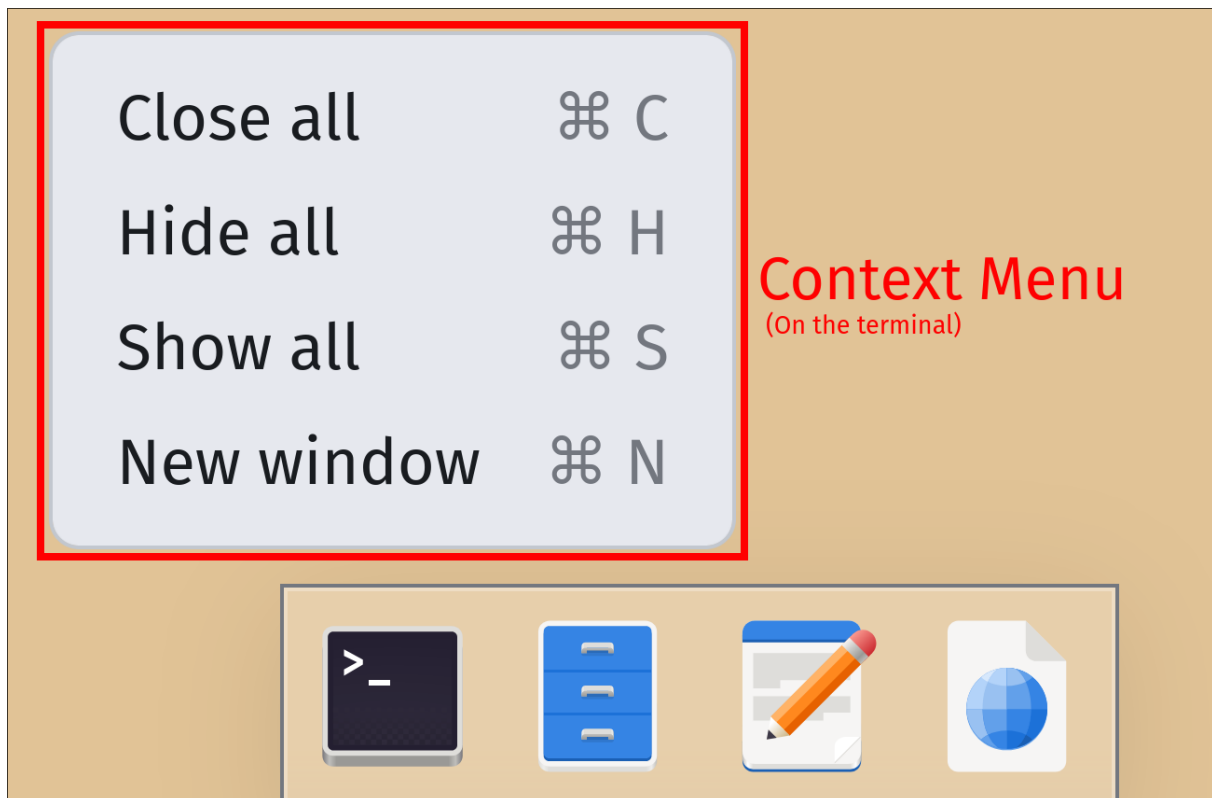
You can “left click” on an icon to open one instance of that application type.

Furthermore, if an instance of this application type already exists when you “left click” them, it will cycle bringing the instances to the forefront, starting with the closest going to the furthest.

Hako comes with four built-in graphical user interface (GUI) applications:

- **Terminal** (first icon)
- **File** (second icon)
- **Text Editor** (third icon)
- **System API Documentation** (fourth icon)

The functionality of these can be read about below in the “Applications” section.



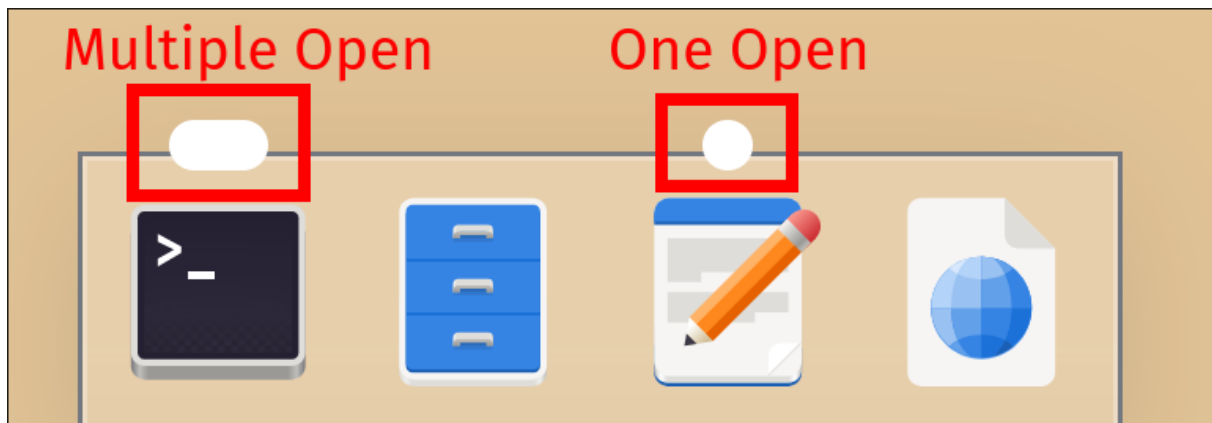
You can “right click” on any application icon to open the context menu.

The context menu allows you to perform actions on a single application type, these actions include:

- **Close all** – Closes all instances of that application type
- **Hide all** – Hides all instances of that application type
- **Show all** – Shows all instances of that application type
- **New window** – Creates a new instance of that application type

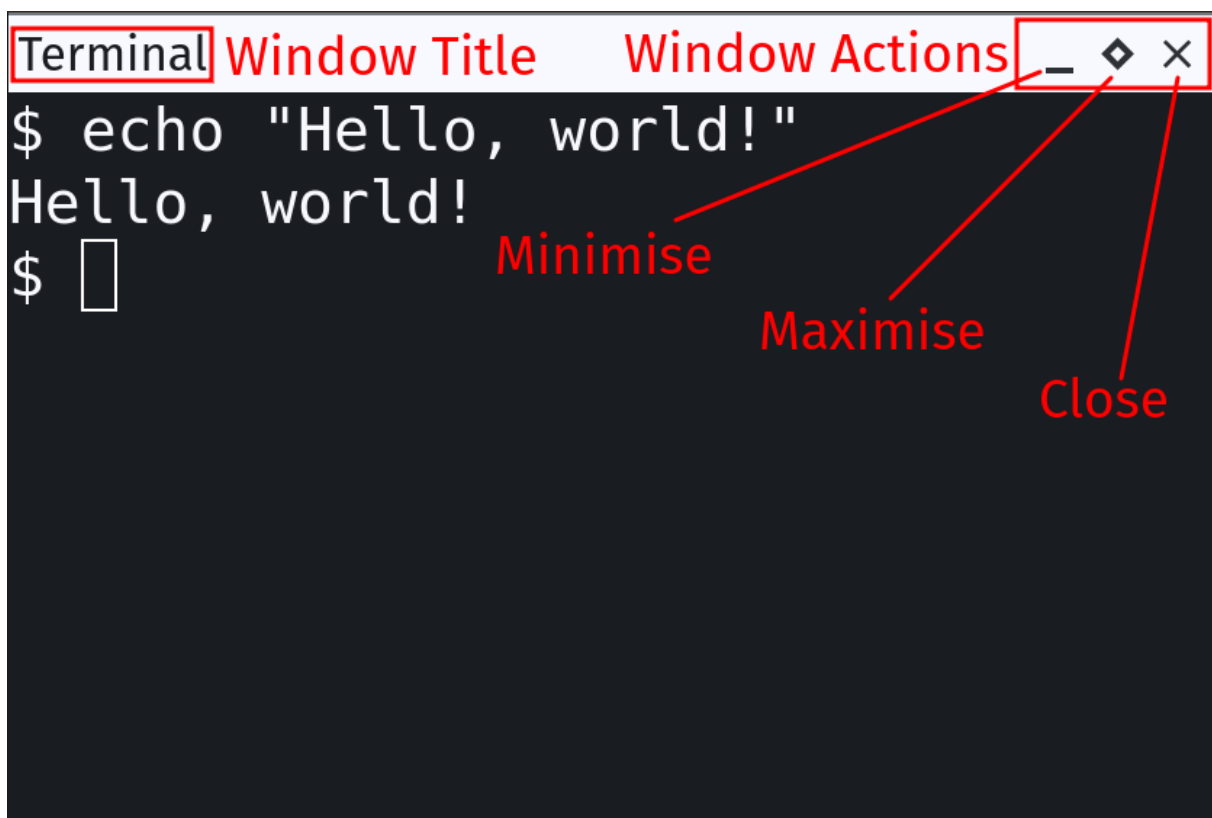
You can also hit a keybind whilst the context menu is open to perform the associated action, as denoted above.

- **C** – Close all
- **H** – Hide all
- **S** – Show all
- **N** – New window



The taskbar also denotes if none, one or multiple instances of an application type are open. See image above.

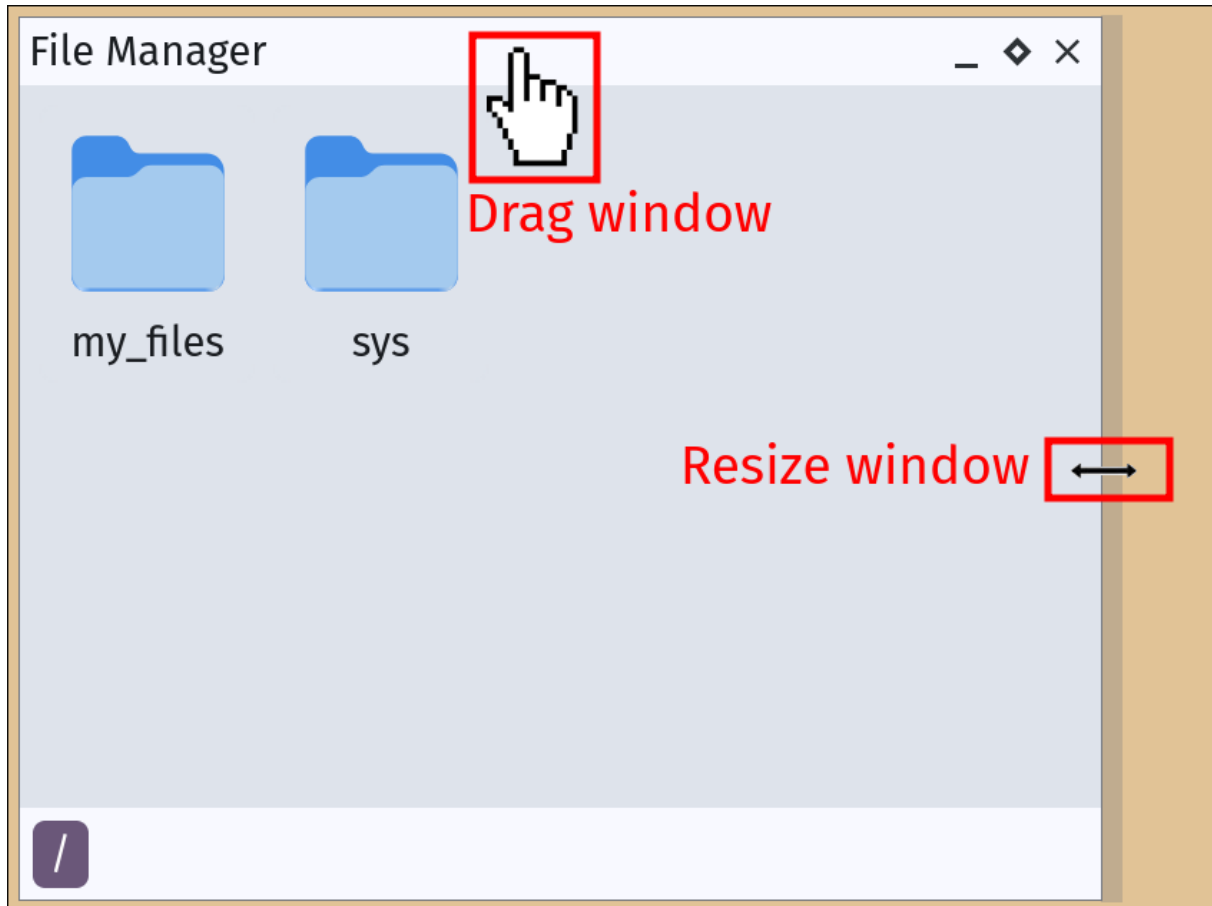
Windows



Windows contain the application's graphical interface, information about the application, and methods to manipulate the application instance.

The window title in the top-left corner displays the window's application's name.

The window actions in the top-right corner allow you to minimise, maximise and close the window.

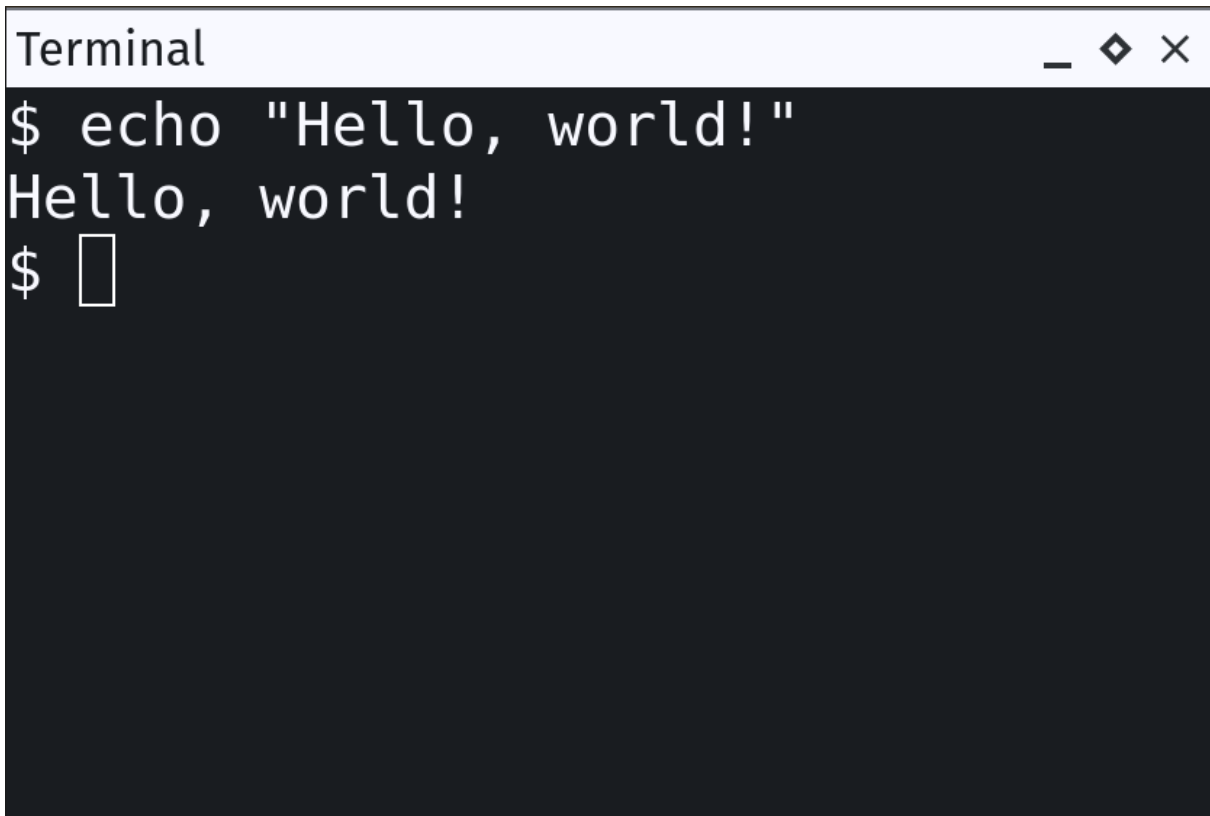


Windows can be dragged around the desktop by holding “left click” on the window's title bar.

Windows can be resized by hovering on any of the window's four borders, and holding “left click” whilst dragging to increase or decrease in the direction you wish.

Applications

Terminal



```
Terminal
$ echo "Hello, world!"
Hello, world!
$ 
```

The terminal is an application that provides a text-based interface that lets you interact directly with the Hako operating system or other programs.

Through a terminal you can:

- **Run commands** to control the system (like managing files, processes, windows, etc)
- **Automate tasks** by writing and running lua scripts
- **Interact with programs** that don't have a graphical user interface (GUI)

The default program that the terminal runs is the **shell**, which you can read more about below in the “core-utils” section.

The terminal also has additional features, such as:

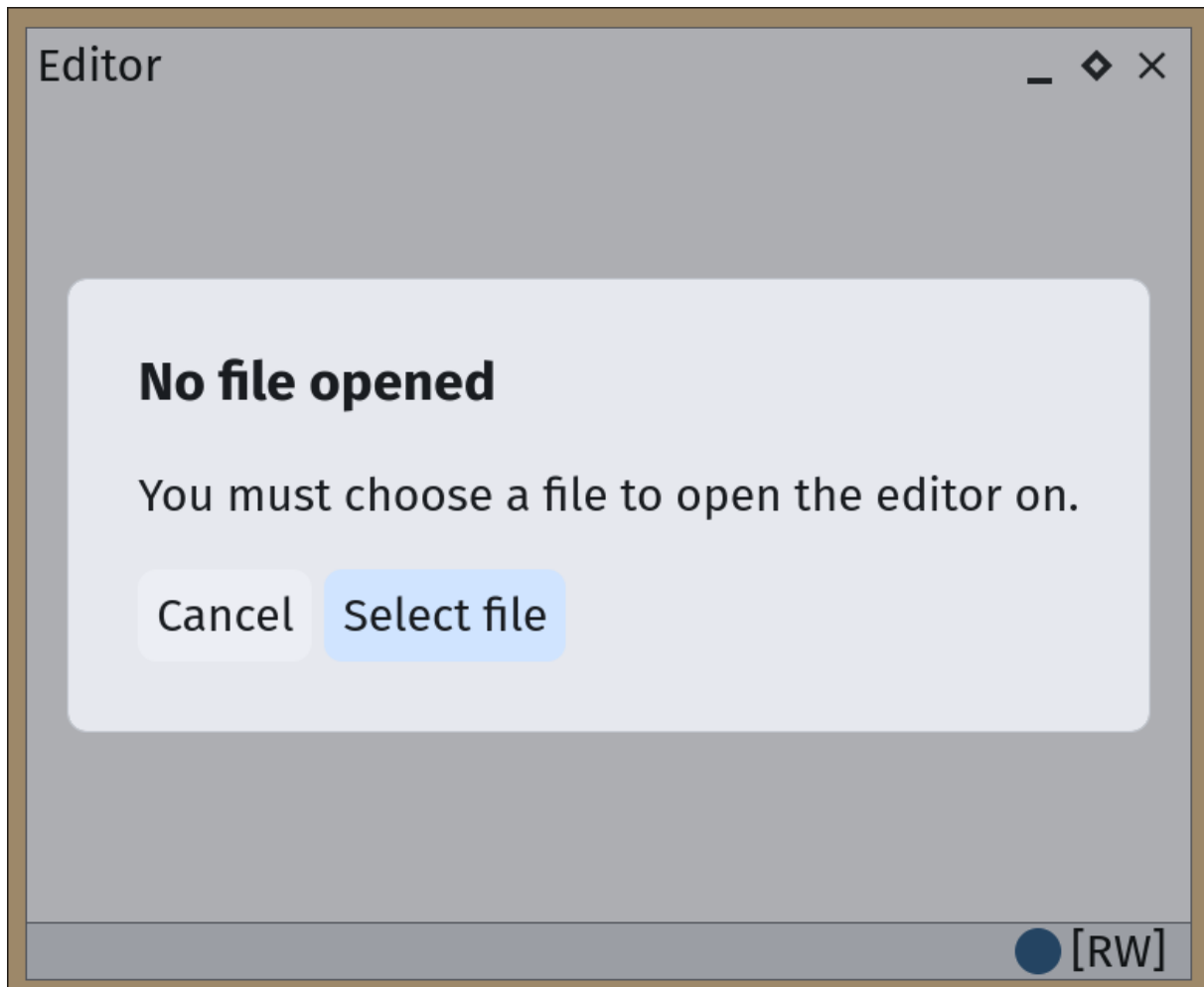
- **Line manipulation** – You can edit a line like you typically would, going back and forth with arrows keys
- **Command history** – Go to previous commands ran with arrows keys
- **Command search** – Enter “Ctrl + R” to search your history for previous commands

If the shell process attached to a terminal dies, the terminal instance will close.

Editor The text editor is a lightweight and fast application that lets you to edit and view plain text easily.

It's primary features are:

- Allow file editing
- Syntax highlighting for Lua files
- Auto-save to avoid loss of data



When you open the text editor from the task bar, it will request a file to open.

Upon clicking “Select file”, it will open the file browser, allowing you to select or create a file.

```
Editing /myshell.lua File being edited
444 -- Grammar of what we're parsing:
445 --
446 -- Lines := Line (';' Line)*
447 --
448 -- Line := Pipeline ( ('&&' | '||') Pipeline )
449 --
450 -- Pipeline := Command ( '|' Command )*
451 --
452 -- Command := SimpleCommand
453 --           | GroupedCommand
454 --
455 -- GroupedCommand := '{' Lines '}'
456 --
457 -- SimpleCommand := word ( word )* Redirects?
458 --
459 -- Redirects := RedirectIn RedirectOut
```

The editor window's name reflects the file being edited.

There are two indicators in the bottom right:

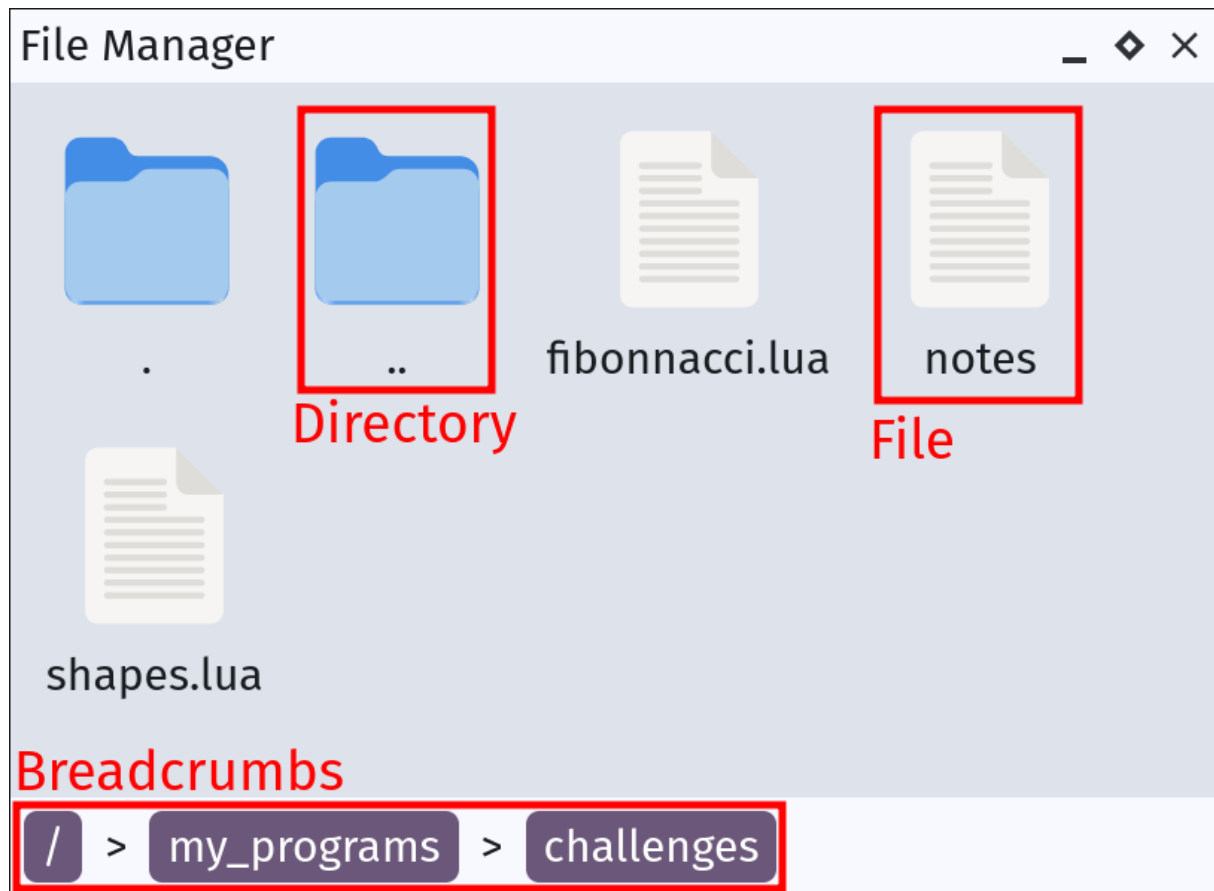
- **Save indicator** – The circle. Blue indicates saved, Red indicates not-saved.
- **File Permissions** – Whether the file is Read Only [RO] or Read Write [RW]

File Browser The file browser is an application that provides a graphical interface for you to view, organise and manage the files and folders on Hako.

The primary features of the file browser are:

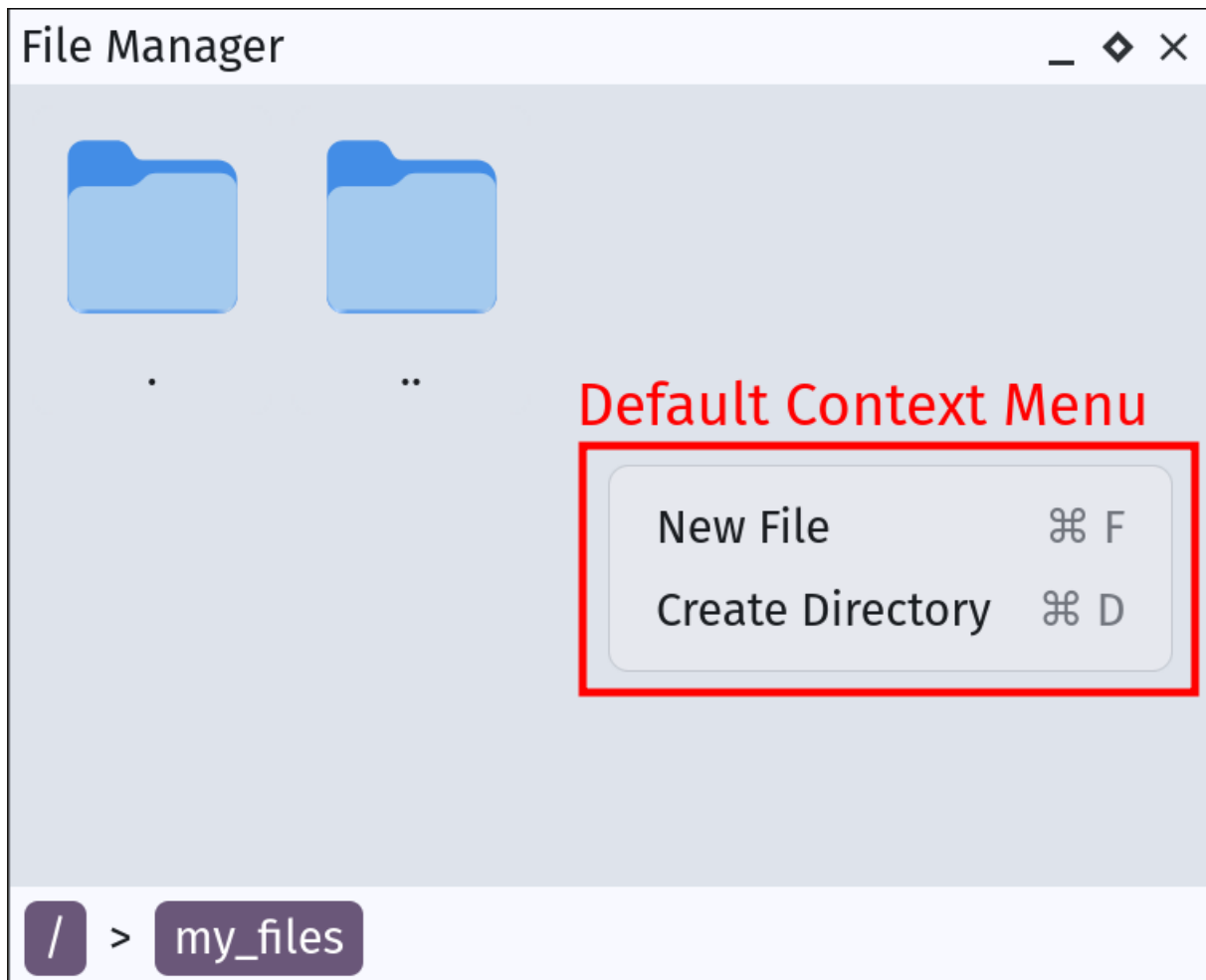
- Viewing your filesystem
- Traversing your filesystem
- Creating files or directories
- Renaming files or directories
- Deleting files or directories

- Drag and drop for moving files or directories



Directories have a “folder” icon. Files have a “notes” icon.

The bottom-left contains “breadcrumbs”, which shows your relative path from the root.

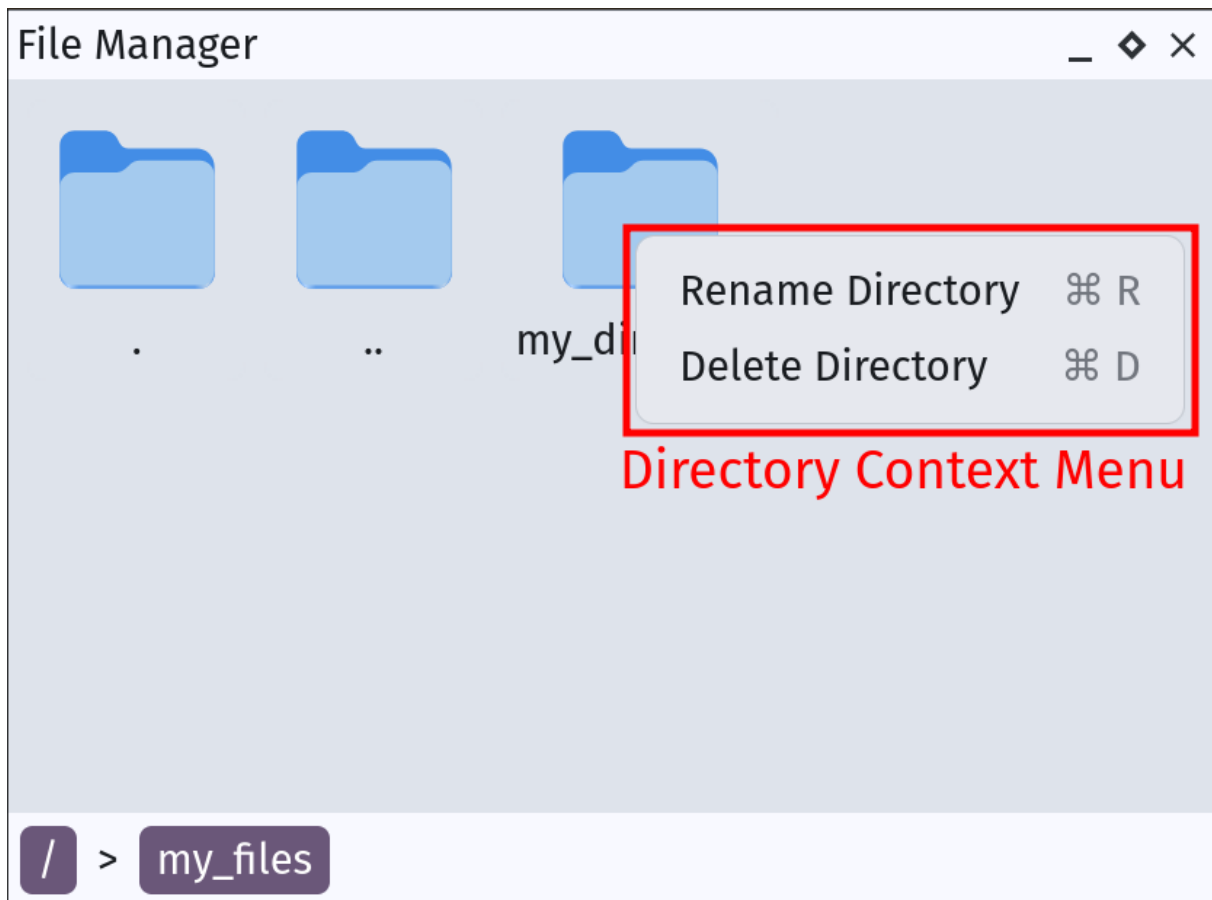


You can “right-click” on the background of the file browser to open the general context menu.

This provides you with two options:

- **New file** – Creates a new file, prompts you for the name
- **Create directory** – Creates a new directory, prompts you for the name

You can also use the keybinds denoted in the context menu instead of directly clicking the options.

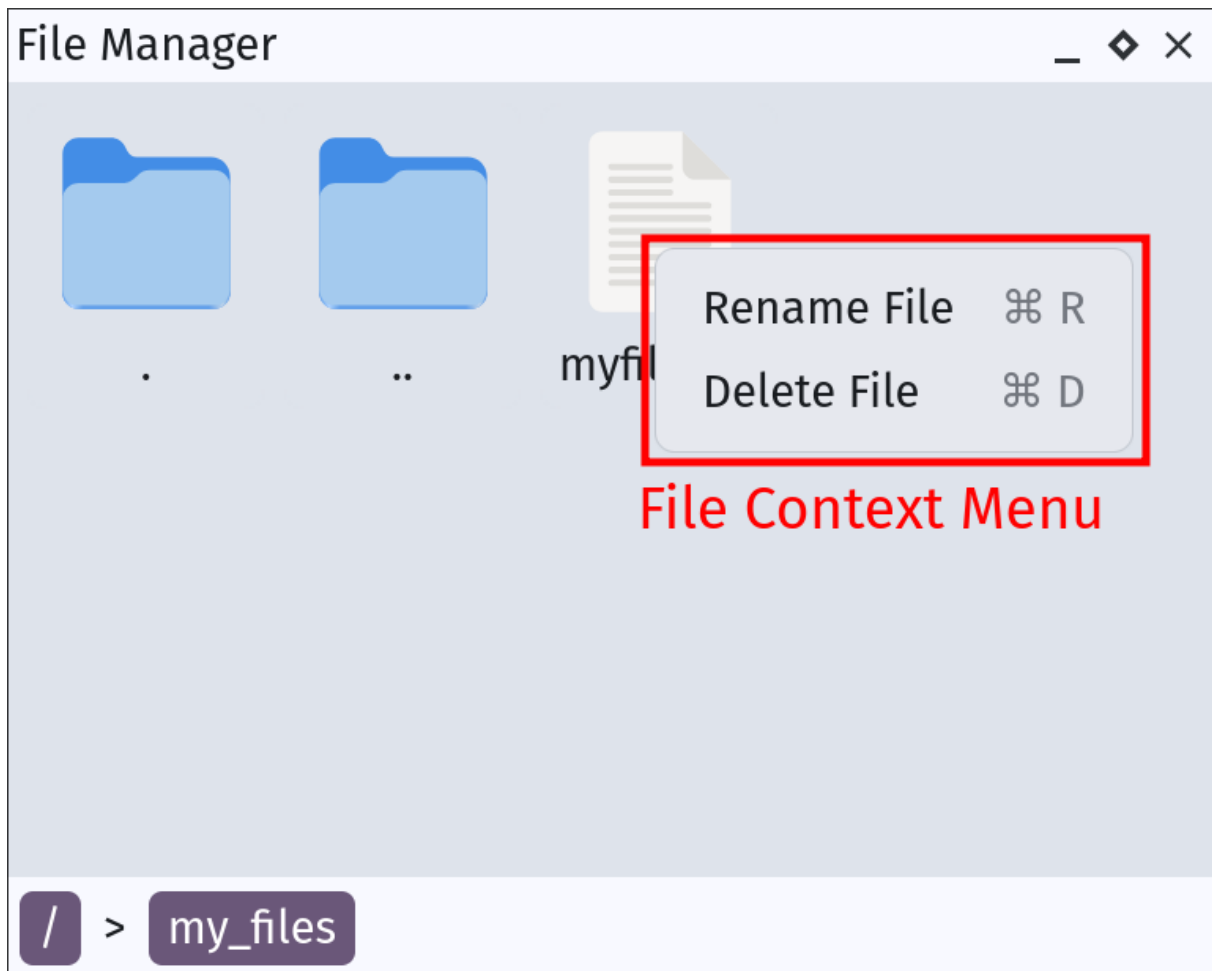


You can “right-click” on a directory to open the directory context menu.

This provides you with two options:

- **Rename directory** – Rename the directory
- **Delete directory** – Delete the directory

You can also use the keybinds denoted in the context menu instead of directly clicking the options.



You can “right-click” on a file to open the file context menu.

This provides you with two options:

- **Rename file** – Rename the file
- **Delete file** – Delete the file

You can also use the keybinds denoted in the context menu instead of directly clicking the options.

Manual

Manual			
Idoc Contents Functions Tables Fields Modules api		Module api	
		Functions	
		file.open (path, flags)	Create or open a file.
		file.close (fd)	Close a file.
		file.write (fd, text)	Write some text to a file.
		file.read (fd, amt)	Read some text from a file.
		file.read_all (fd)	Read all the text from a file.
		file.shift (fd, amount)	Move the cursor for open file forward.
		file.jump (fd, position)	Move the cursor to position in open file.
		file.remove (path)	Remove a file.
		file.move (old_path, new_path)	Move file or directory from one path to another (rename).
		file.make_dir (path)	Make a directory.
		file.remove_dir (path)	Remove a directory.
		file.change_dir (path)	Change the current working directory.
		file.read_dir (path)	Read the contents of a directory (akin to 'ls').

Hako has multiple system APIs, described below in the “System APIs” section.

The manual is a full document describing all the different ways to interact with Hako.

The alternative is the [help](#) command that you can enter into your terminal, which describes Hako’s system APIs at a higher level with examples

Core-Utils

Shell AND OTHERS

System APIs

Global Constants

Desktop Environment API (window)

Process API (process)

Filesystem API (file)

Terminal API (terminal)

Format API (fmt)

Error API (errors)