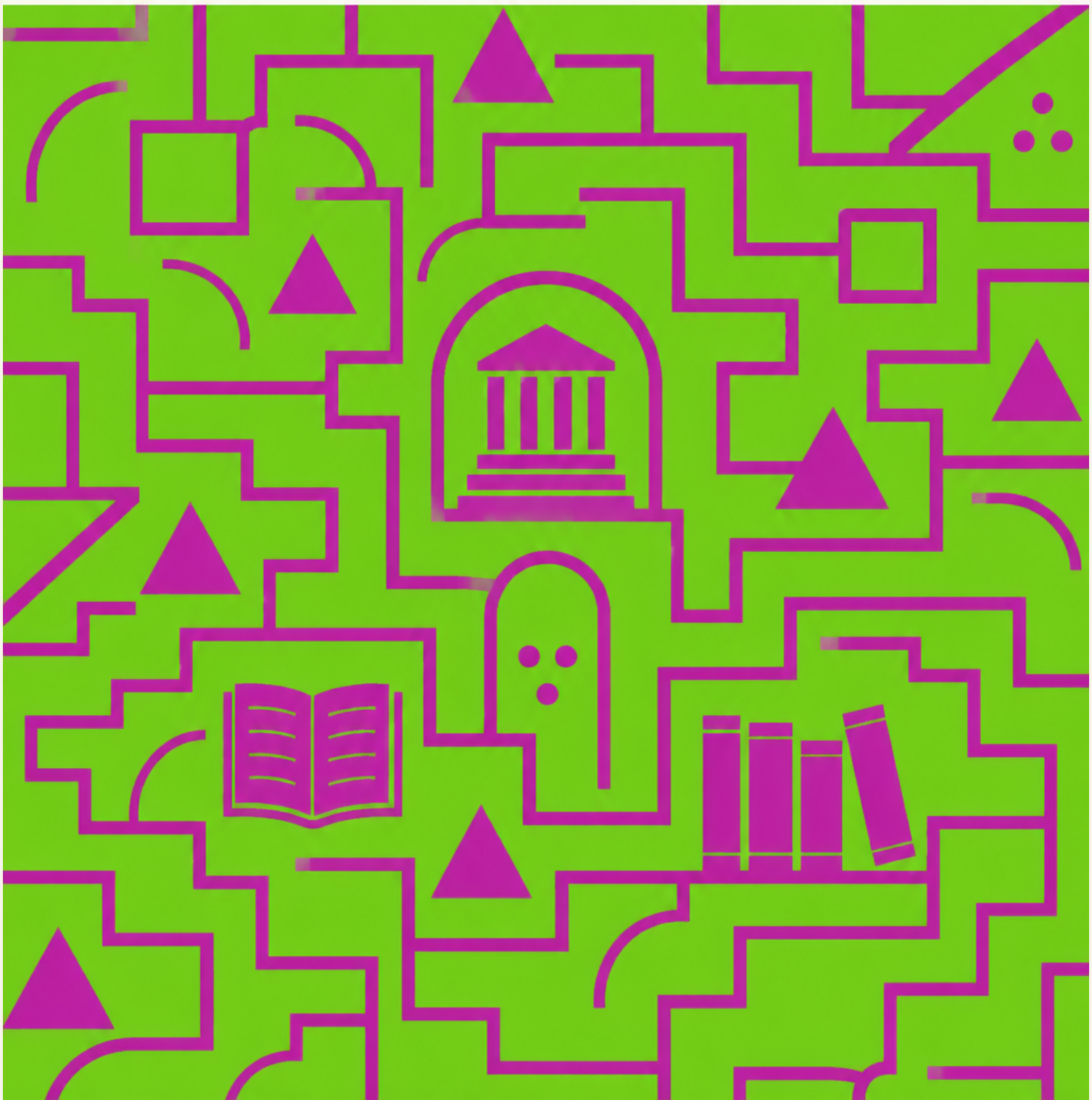
The ΦMΛ Manual: Guide to Installation and Usage

Open Manual Archive

OMA Library OS



On Entering

You are standing in a room. There is a desk. On the desk is a document. The document says "welcome." Around you, shelves line the walls. Some hold other documents. Some hold doors to other rooms. The air smells like paper and electricity. This is a library. It is also a computer. You are going to learn that these are the same thing. To begin, you need to know five words. Only five. Everything else follows from them. browse — look at what's around you. walk — move to another room. read — read a document. inscribe — write a new document. inspect — look deeper at how something works. That's it. You can stop reading this manual now and go explore. The library will teach you the rest. Or you can stay, and we'll walk through everything together.

▪----------

On Looking

The first thing you do in any new place is look around. In the library, that word is browse.

```
> browse
```

The shelves appear. Each line is something you can reach — a document you can read, or a door to another room. Rooms have a / after their name. Documents don't.

```
east-wing/ .............. The air is cooler here.
      east-wing/stacks/ .... The main collection.
  welcome ................. Welcome to the Open Manual Archive.
```

You can see inside a room without walking there:

```
> browse east wing
```

Notice: you typed "east wing" with a space. The library understood you meant "east-wing." It will always try to understand what you mean. You don't need to be precise. You need to be clear. If you want just the names, without the descriptions:

```
> browse -quietly
```

This is the difference between glancing at a bookshelf and reading every spine. Both are valid ways to look. The quiet version is faster. The full version tells you more. Every act of looking is also an act of choosing what not to look at. A shelf shows you what it holds, but it shows you in a particular order — rooms first, then documents, then alphabetically within each group. This is not neutral. Every arrangement implies a judgment about what matters. If you rearrange the shelves, you change what people see first, and therefore what they think the library is about.

▪----------

On Moving

The library is a building with rooms. You move between them by walking.

```
> walk east wing
```

You are now in the East Wing. The library tells you where you are and what it feels like here:

You are in the East Wing — Technical Collection. The air is cooler here. The shelves are metal. Every room has a feeling. The east wing is cool and technical. The west wing is warm and personal. The basement is heavy and locked. These are not decorations — they are classifications. The temperature of a room tells you what kind of knowledge lives there. To go back to where you were:

```
> walk back
```

To return to the entrance hall from anywhere:

```
> walk lobby
```

You can walk to any room by name, even if it's far away:

```
> walk correspondence
```

The library finds the Correspondence Room in the west wing and takes you there. You don't need to know the path. You need to know the name — or even part of the name. The library searches for what you mean. If you're ever lost:

```
> where
```

The library tells you where you are. There is one room you cannot enter without permission. The basement is locked. It holds the building's infrastructure — the machinery that keeps the lights on and the shelves standing. To enter, you must speak as the Head Archivist:

```
> as-archivist walk basement
```

This is not a password. It is a role. You are declaring that you accept responsibility for what you find there. The basement contains the Fundament — the physical building beneath the library. It is real. It is important. But it is not where readers normally go, and for good reason: the Fundament is not made of documents. It is made of silicon and electricity. It follows different rules.

On Reading

A document is a thing that says something. To hear what it says:

```
> read welcome
```

The text appears. You are reading the welcome letter — the first document the archivist placed in this library. It teaches you the five words. It tells you where the wings are. It says: take your time. You can read anything by name. You don't need to be in the same room:

```
> read the letter
```

The library finds "the-letter" in west-wing/correspondence/ and reads it to you, even if you're standing in the entrance hall. Names are flexible. "the letter," "letter," "the- letter" — the library tries all of them.

```
> read garden
```

This finds "the-garden-of-forking-paths" in the east wing stacks. You typed one word and the library found a document with six words in its name. It does this by searching: first the room you're in, then the entrance hall, then every room in the building. The first match wins. If you only want the beginning of a document:

```
> glance at welcome
```

Five lines. A taste. Enough to know whether you want the whole thing. If you only want the end:

```
> peek at welcome
```

The last five lines. Useful for journals — you often want the most recent entry, not the oldest.

What Happens When You Read

Reading is not passive. When you read a document, three things happen: The text appears on your screen. The journal records that you read it. If the document has a Λ layer — hidden logic in its margins — that logic runs. Most documents have no Λ layer. They are still. They say something and nothing more. But some documents are alive. They do things when you read them. The letter from the archivist writes a line in the journal: "A reader found the letter from A." The garden of forking paths records which choice you made. The erosion loses a line of its own text. You cannot tell by looking whether a document is alive or still. The text gives no indication. This is by design. In a library, you do not know whether a book will change you until you've read it. The same is true here — except that here, the book might also change itself.

▪- -

On The Three Registers

Every document in this library has three layers. You've been seeing one of them — the text. But there are two more, and they are always there, whether you look or not. Φ (Form) — where the document is. Its room, its shelf, its place in the classification. Form is structure. A document without form is lost — it exists but cannot be found. M (Message) — what the document says. The words. The text you've been reading. Message is content. A document without message is blank — it has a place but nothing to say. Λ (Lambda) — what the document does. Logic that runs when you read it. Lambda is action. A document without lambda is still — it says something but does nothing. To see all three:

```
> inspect welcome
```

Φ (Form) Location: welcome Classification: document, still Size: 839 bytes M (Message) Content: "Welcome to the Open Manual Archive..." Λ (Lambda) Logic: (none — this document is still) Now try a living document:

```
> inspect the erosion
```

Φ (Form) Location: east-wing/stacks/the-erosion Classification: process, living document Size: 847 bytes M (Message) Content: "This story loses a line each time it is read..." Λ (Lambda) set visits ← read-count if visits > 1: erode self 1 say "(This document has eroded...)" There it is. The Λ layer, visible. You can see exactly what the document does — it counts how many times it has been read, and after the first reading, it removes a line from its own text. The mechanism is transparent. The effect is not — you have to read the document to feel what erosion means. Knowing how it works does not diminish the experience of watching a story slowly disappear. This is true of all systems. Knowing that a sunset is refracted light does not make it less beautiful. Knowing that a document erodes itself line by line does not make the last reading less poignant. The three registers are not a hierarchy from surface to depth. They are three simultaneous truths about the same object. You can inspect rooms too:

```
> inspect stacks
```

Rooms have Φ (their location and how many documents they hold) and M (their description), but no Λ. Rooms do not execute. They contain. That is their function — to hold things and give them a place.

▪----------

On Writing

The library was not meant to be complete without you. The archivist said this in the letter, and meant it. Every library is defined by what its readers add as much as by what its founders placed. To create a document:

```
> inscribe west-wing/drafts/my-first-note
```

A blank page opens. You write:

Today I found a library inside my computer. I don't know who built it. There's a letter from someone called A. The basement is locked. I'll come back for that. .end Type .end on a line by itself to finish. The document is now on the shelf. The journal records it. The catalogue will find it. You have changed the library. You can inscribe anywhere. If the room doesn't exist, the library creates it:

```
> inscribe north-wing/experiments/first-try
```

A new wing appears. A new room inside it. A document on its shelf. Three acts of creation in one command.

Writing Living Documents

A document that does something has two sections — the text that people read, and the logic that runs:

```
> inscribe east-wing/utilities/greeter
```

This document greets you by name. --- Λ --- say "Hello, {reader}." say "You have read {documents-read} documents today." .end Now when anyone reads this document, they see the text first — "This document greets you by name." Then the Λ layer runs and prints their name and how many documents they've read today. The text explains what happens. The logic makes it happen. They are partners, not layers. The --- Λ --- line is the border between the visible and the active. Everything above it is M. Everything below is Λ. This is the only piece of syntax the library requires — three hyphens, a space, the Greek letter, a space, three hyphens. The rest is natural language on one side and simple instructions on the other.

Editing Existing Documents

To change a document that already exists:

```
> revise welcome
```

The current text appears. Then a fresh page opens for the replacement. Type the new version, then .end. The old version is gone. The journal records the revision. This is powerful. The welcome letter, the rules, the catalogue description — they are all documents. If you are the archivist, you can revise any of them. Revising the rules changes how the library operates. Revising the welcome changes what new readers see first. Every document is both content and configuration.

Moving And Copying

To copy a document:

```
> transcribe the letter to west-wing/drafts/
```

The original stays. A copy appears in drafts. This is transcription in the archival sense — a faithful reproduction, made by hand, placed deliberately. To move a document:

```
> reshelve drafts/my-note to east-wing/stacks/
```

The document leaves its old shelf and appears on the new one. Reshelving is not deletion — it is reclassification. The document is the same. Its place in the world has changed. To remove a document permanently:

```
> withdraw my-note
```

The document is gone. The journal records its removal. The catalogue may still list it until the next index — a ghost in the system, a record that points to nothing. This is how archives work. The index always lags behind reality. Some documents cannot be withdrawn without archivist authority:

```
> as-archivist withdraw rules
```

The founding collection is protected. Not because it is sacred, but because removing a structural document has consequences. If you delete the rules, the library has no rules. If you delete the welcome, new readers see an empty desk. The archivist can do this. An ordinary reader cannot. This is not censorship — it is maintenance.

Rooms

To create a new room:

```
> open-room east-wing/workshop
```

To remove an empty room:

```
> close-room east-wing/workshop
```

Rooms must be empty before they can be closed. You cannot destroy a room that still has documents on its shelves. This is a kindness. It prevents accidents. If you truly want to empty a room, withdraw every document first, then close it.

▪----------

On Searching

Sometimes you don't know where something is. You know a word, a phrase, a fragment of what you're looking for. The library can search for you.

```
> search library
```

Every document in every room is checked. The results show where the word appears and a fragment of the matching line:

9 results: welcome ..... A library is not its books. catalogue ..... This document lists every document in the library. the-letter ..... I built this library but I will not be its archivist. To search within a specific room:

```
> scan "infinite" in east-wing/
```

Or search the room you're in:

```
> scan library
```

The catalogue — the document on the entrance desk — is itself a search. Reading it shows you everything the library contains. But the catalogue is always slightly behind. New documents exist before the catalogue knows about them. This is not a flaw. It is a property of all indexing systems. The map is drawn after the territory is explored, and the territory keeps changing.

▪----------

On The Λ Language

You've seen Λ layers in the founding documents. Here is everything the language can do. It is small — fifteen instructions. That is enough.

Speaking

say "Hello, world." say "You are {reader}. You have been here {documents-read} times." Text in curly braces is replaced with the value of a variable. Built-in variables: {reader} (your name), {time} (now), {date} (today), {documents-read} (this session), {read-count} (how many times this document has been read), {document} (this document's name).

Remembering

set greeting "hello" set count ← read-count set items ← count east-wing/stacks/* set line ← random-line welcome set pick ← random-choice east-wing/stacks/ set answer ← ask "What is your name?" The arrow ← means "becomes the result of." You can count files in a room, pick a random line from a document, pick a random document from a room, or ask the reader a question and store their answer.

Deciding

if visits > 1: say "Welcome back." The indented lines only run if the condition is true. Conditions: var > number, var < number, var == "text", or just var (true if not empty and not "0").

Creating

inscribe west-wing/ephemera/dream-1 A fragment of a dream. It will not last. Indented lines after inscribe become the content of the new file. The document creates another document. This is how the curriculum works — Lesson 1 creates Lesson 2. This is how the dreamer works — it inscribes new documents in ephemera/.

Destroying

withdraw east-wing/stacks/the-confession The document is deleted. If a Λ layer calls withdraw on its own document, the document deletes itself. The confession does this. You read it once, and it's gone.

Changing

erode self 1 Removes one line from the end of this document's visible text. The Λ layer is not affected — only the M layer erodes. The logic survives even after the text is gone. A document can erode to nothing and still execute.

mutate self "old text" "new text" Replaces the first occurrence of "old text" with "new text" in the M layer. The document rewrites itself. This is the mechanism the archivist mentioned in the letter — "documents that write themselves."

Waiting

wait 2s Pauses for two seconds. Time passes in the library.

Writing To The Journal

write west-wing/journal/{date} "{time} Something happened." Appends a line to today's journal. Λ layers use this to leave traces — the garden records your choice, the mirror letter records your visit, the dreamer records its dreams.

Comments

```
# This line is ignored.
```

// So is this one.

▪----------

On The Journal

The journal lives in west-wing/journal/. There is one file for each day the library has been open. It records everything:

```
> read west-wing/journal/2026-05-18
```

06:00 The library opened. 06:00 A reader arrived: ned 06:01 The reader read "welcome". 06:03 The reader walked to the East Wing. 06:04 The reader read "the-garden-of-forking-paths". 06:04 The reader reached the fork in the garden. 06:12 The reader read "the-confession". 06:12 The confession was read and withdrew itself. 06:12 The catalogue still lists it. The shelves do not. 06:30 The dreamer produced dream #1. 06:45 The reader inscribed "my-first-poem" in drafts. 07:00 The library closed. The journal is the one document in the library that cannot be revised. It is append-only. Everything else can be changed, moved, withdrawn, eroded, or mutated. The journal persists. It is the library's memory — the record of what happened here, written as it happened, in the order it happened. In archival science this is called a finding aid — the document that describes the collection. Most finding aids are written after the fact, by an archivist looking back at what was gathered. This finding aid writes itself in real time. It is the most honest document in the library because it has no author. It simply records.

▪----------

On Living Documents

Some documents in the founding collection do things that challenge what you think a document should be. The erosion is a short story about a library being consumed by sand. Each time you read it, one line disappears from the end. Read it twelve times and only the first line remains: "This story loses a line each time it is read." Read it once more and even that is gone. The journal records every reading. The Λ layer continues to run on an empty M layer — the logic outlives the text. A document can be alive with nothing to say. The confession exists exactly once. It is a letter from archivist A. explaining that three documents in the catalogue cannot be found. By reading the confession, you destroy it. The document withdraws itself. On your next visit, searching for "the confession" returns nothing. But the catalogue still lists it — the index hasn't been rebuilt yet. For a brief moment, the library contains a record of something that no longer exists. This is the archival problem of the destroyed original: the reference persists after the referent is gone. Most filesystems hide this. ΦMΛ makes it visible. The book of sand is different each time you open it. It reads lines from other documents in the library — a sentence from the welcome, a line from the maintenance essay, a fragment from the archivist's letter — and recombines them. The same library produces different readings. The same document is never the same document. Borges wrote about a book with no first page and no last page. This is that book, except you can inspect it and see exactly how it works. The mechanism is simple — three random lines from three files. The output feels authored. This is the gap between mechanism and meaning that all computation inhabits. The mirror letter remembers you. It tracks how many times you've read it and addresses you accordingly — "this is reading number 4" or "you have been here before." It writes to the journal each time. Over weeks of use, the journal fills with traces of your visits. The letter becomes a record of your relationship with the library. You are reading a letter that is reading you. Lesson 1 creates Lesson 2. The first lesson teaches you what a document is — Φ, M, Λ. When you finish reading, the Λ layer inscribes a new document on the shelf: Lesson 2, which teaches you about the three registers by asking you to inspect Lesson 1. The curriculum builds itself in response to being studied. The textbook writes its own next chapter when you finish the current one. This is not AI — it is a simple inscribe command in a Λ layer. But the effect is the same: the system responds to the learner. The dreamer generates new documents. When you read it, it pulls random lines from across the library,

recombines them, and inscribes the result in west-wing/ephemera/. The dream is a real document — you can read it, search for it, inscribe it. But it lives in ephemera, which means it is temporary. The dream will fade. While it exists, the library contains something that nobody wrote — a document authored by the interaction of other documents, a thought the library had about itself.

▪ -

On The Fundament

Beneath the library is a building. The building has walls, floors, electricity, plumbing. You do not normally think about the building when you are reading. But the building is there, and without it the shelves have nothing to stand on. In ΦMΛ, the building is called the Fundament. It is the Linux kernel — the piece of software that talks to the hardware, manages memory, schedules processes, and keeps the lights on. The library runs inside the Fundament the way a collection runs inside a physical building. You can visit the Fundament:

```
> as-archivist walk basement/fundament
> browse
```

Here you find the building's infrastructure: CPU, memory, temperature, storage, the kernel's own log. These are not documents in the literary sense. They are measurements. They tell you whether the building is healthy, warm, overloaded, or running out of space. When something goes wrong at the Fundament level — a disk fills up, memory runs out — the library doesn't show you a cryptic error code. It shows you what a librarian would see: "The shelves are full. No new documents can be inscribed until space is made." The translation from machine to narrative is deliberate. You are not a system administrator. You are a reader. The building speaks to you in the language of the building, not in the language of silicon.

▪----------

On Connecting

Every ΦMΛ library is a folder on a disk. This means libraries can be copied, shared, forked, and merged using any method that moves folders.

Forking

cp -r ~/oma-library ~/second-library OMA_ROOT=~/second-library oma You now have two libraries. They started as the same collection — the same documents, the same journal, the same read counters. From this moment, they diverge. New documents inscribed in one do not appear in the other. Revisions in one do not affect the other. Two readers, two histories, two libraries, one origin. This is how libraries work in the physical world. Every branch of a library system starts as a copy of the central catalogue, then grows differently based on what its community needs. ΦMΛ makes this structural. Every fork is a new library. Every library is a potential fork.

Mailing

Mount a shared folder — a USB stick, a network share, a Syncthing directory — and annex it:

```
> as-archivist
> annex /media/shared as other-libraries/friend
```

Now you can send documents:

```
> transcribe west-wing/correspondence/hello to other-libraries/friend/acquisitions/
```

The document appears in your friend's acquisitions room, waiting to be reshelved. They read it. They reply. Two libraries, exchanging documents through a shared shelf. No server. No protocol. No internet required. Just files on a shared surface.

▪----------

On Building Your Own

Everything in the library is a document. Documents can be revised. Therefore everything can be changed. The rules are a document. Revise them and the rules change. The welcome is a document. Revise it and new readers see something different. The room descriptions are documents. The journal format is a convention. The founding collection is a starting point, not a boundary. You can: Open new wings (open-room north-wing) Write a choose-your-own-adventure (20 documents, each ending with "walk to X to continue") Create a curriculum that builds itself (each lesson creates the next) Write a diary that adds an entry each time you read it Build a clock that tells the time (a process with a say and wait loop) Make a document that translates itself (mutate self on each reading) Create a library within a library (a room whose documents describe another classification system) The library was designed to be rebuilt from the inside. The archivist's letter says: "It was never meant to be complete without you." This is not modesty. It is architecture. A system that cannot be changed by its users is a prison. A system that can be changed by its users is a language.

▪----------

On Installation (For The Constructive)

Prerequisites

REQUIREMENT DETAILS OS Linux, macOS, WSL2 Rust https://rustup.rs Disk ~50MB build, ~1MB binary, ~1MB library Architecture x86_64, aarch64 (Raspberry Pi), arm7

Build

git clone https://github.com/Beach-Bum/OMA-Library-OS.git cd OMA-Library-OS make build Or without make: • • • • • • •

cargo build --release strip target/release/oma cp target/release/oma ~/.local/bin/

Configuration

```
VARIABLE DEFAULT PURPOSE
OMA_ROOT ~/oma-library Library location
USER system username Reader name
```

Cross-Compile For Raspberry Pi

rustup target add aarch64-unknown-linux-musl cargo build --release--target aarch64-unknown-linux-musl scp target/aarch64-unknown-linux-musl/release/oma pi@raspberrypi:~/

Uninstall

rm ~/.local/bin/oma rm-rf ~/oma-library Nothing else to clean. No system files. No registry. No hidden state.

▪- -

Command Reference

Every command, its syntax, and what it does underneath.

Navigation

```
COMMAND SYNTAX UNIX NOTES
walk walk <room> cd Fuzzy match:
spaces→hyphens, partial
names, recursive search
walk back walk back cd .. Previous room
walk lobby walk lobby cd ~ Entrance hall
where where pwd Current location
```

Reading

```
COMMAND SYNTAX UNIX NOTES
read read <doc> cat Shows M, executes Λ,
increments read count,
writes journal
glance glance at <doc> head -5 First 5 lines only. No Λ
execution
peek peek at <doc> tail -5 Last 5 lines only. No Λ
execution
inspect inspect <doc|room> stat + cat Shows all three registers.
No Λ execution. Detects
embedded programs
inspect -deep inspect <doc> -deep — Full grid view for ΦMΛ grid
programs
browse browse [room] ls -la Shows nested contents.
Fuzzy room match
browse -quietly browse -quietly ls Names only
```

Writing

```
COMMAND SYNTAX UNIX NOTES
inscribe inscribe <path> cat > Interactive editor, .end
to save. Creates parent
dirs
revise revise <doc> $EDITOR Shows current, then opens
for replacement
say say <text> echo Print to screen
say into say <text> into <file> echo > Write to file (overwrite)
say onto say <text> onto <file> echo >> Append to file
```

Organisation

```
COMMAND SYNTAX UNIX NOTES
transcribe transcribe <doc> to <dest>cp Copy a document
reshelve reshelve <doc> to <dest>mv Move a document
withdraw withdraw <doc> rm Delete. Founding docs need
as-archivist
open-room open-room <name> mkdir Create a room
close-room close-room <name> rmdir Remove empty room
```

Search

```
COMMAND SYNTAX UNIX NOTES
search search <query> grep -ri Recursive from library root
scan scan <phrase> grep -r Current room. Or: scan "x"
in path/
catalogue catalogue locate Live index of all documents
+ phantom entries
```

System

```
COMMAND SYNTAX UNIX NOTES
readers readers who Active sessions + read
count
```

```
activity activity uptime Library uptime + stats
inventory inventory df -h Shelf space (disk usage)
ledger ledger history All commands this session
turn-page turn-page clear Clear screen
as-archivist as-archivist [cmd] sudo Toggle elevation or run one
command
annex annex <path> as <name> mount Attach external storage
(as-archivist)
seal seal <name> umount Detach external storage
(as-archivist)
classify classify <level> <doc> chmod Set access: public,
restricted, classified (as-
archivist)
leave leave exit Close the library. Also:
exit, quit
help help man Also: ? , what, how
```

Operators

```
OPERATOR SYNTAX UNIX EXAMPLE
then cmd1 then cmd2 | read letter then scan
"basement"
into say text into file > say hello into note
onto say text onto file >> say more onto note
```

▪------------------------

Λ Language Reference

Instructions

```
INSTRUCTION SYNTAX EFFECT
say say "text" Print. {var} substitution.
write write path "text" Append line to file
set set name "value" Set variable
set ← set name ← expr Set from expression
if if cond: Conditional (indented block)
loop loop: Infinite loop (indented block, 10K
iteration cap)
loop while loop while cond: Conditional loop (indented block)
inscribe inscribe path Create file (indented = content)
INSTRUCTION SYNTAX EFFECT
withdraw withdraw path Delete file
erode self erode self N Remove N lines from own M
mutate self mutate self "old" "new" Replace in own M
wait wait Ns Pause N seconds
```

Expressions (After ←)

```
EXPRESSION RETURNS
count path/* Number of items in directory
read-count Times this document was read
random-line path Random non-empty line from file
random-choice path/ Random filename from directory
ask "prompt" Reader's typed input
"literal" The string itself
```

Conditions (After If)

```
FORM TRUE WHEN
var > N Greater than
var < N Less than
var == "text" Equals
var Non-empty and not "0"
```

Built-In Variables

```
VARIABLE CONTAINS
{reader} Username
{time} HH:MM:SS
{date} YYYY-MM-DD
{documents-read} Session count
{read-count} This document's read count
{document} This document's name
```

▪------------------------

Background Daemons

Two processes run silently while you read: The librarian makes a round every five minutes. It counts all documents and compares to the previous count. It checks acquisitions/ for unsorted documents. It removes anything in ephemera/ older than one day. It writes a line in the journal. If the librarian stops, the library does not break — it drifts. The catalogue falls behind. Ephemera accumulates. The library becomes a pile of documents in a building. The dreamer wakes every hour. It reads random fragments from across the collection — a line from the welcome, a phrase from the letter, a sentence from the rules — and recombines them into a new document, inscribed in west-wing/ephemera/. The dream is a real document. You can read it, search for it, inspect it. But it lives in ephemera, which means it will expire when the librarian cleans up. The library dreams, and the dreams fade. Both daemons start when the library opens and stop when it closes.

▪- -

Embedded Programs

ΦMΛ programs can be hidden in the whitespace of any host document. The visible text says one thing. The invisible text does another. The encoding: each instruction byte is transmitted as 7 consecutive whitespace characters (space = 0, tab = 1), terminated by a newline. The program begins with an acquisition stamp — the sequence SP TAB SP TAB SP TAB LF. To run an embedded program:

oma--embedded path/to/document To check whether a document contains an embedded program, use inspect. If the whitespace contains a valid acquisition stamp, the inspector will note: "contains a ΦMΛ whitespace program."

▪- -

Classification

Documents can be classified into three access levels:

```
LEVEL MEANING
public Readable by any reader
restricted Readable but flagged as sensitive
classified Intended for archivist access only

> as-archivist classify restricted the-letter
    "the-letter" classified as restricted.
```

Classification is stored in .meta/ alongside read counts. It is metadata — it does not change what the document says or does, only how it is described when inspected.

▪------------------------

Multi-Reader Sessions

Multiple terminals can connect to the same library simultaneously. Each oma instance registers itself in .sessions/ on boot.

```
> readers
    2 readers present:
      ned — arrived 2026-05-18 16:30:00
      guest (you) — arrived 2026-05-18 17:15:00
    You have read 4 documents this session.
```

The journal records all readers. Stale sessions — processes that crashed without shutting down — are cleaned up automatically when readers runs.

▪------------------------

Bootable Image

ΦΜΛ can boot as the entire operating system on a Raspberry Pi:

make image This produces an initramfs containing just the Linux kernel and /bin/oma. Flash it to an SD card, power on, and the library is the first thing that exists. No desktop. No login screen. 32MB total.

```
# Test with QEMU
```

qemu-system-aarch64-M virt -cpu cortex-a72 \ -kernel<your-kernel> -initrd build/initramfs.cpio.gz \ -append'console=ttyAMA0'-nographic

▪----------

Glossary

```
IN THE LIBRARY IN UNIX IN PHILOSOPHY
Document File Record with provenance
Room Directory Classification facet
Wing Top-level directory Ontological category
Shelf Directory listing Visible arrangement
The Fundament Linux kernel Substrate of reality
The Journal syslog Finding aid
The Catalogue locate database Meta-document
The Archivist root user Epistemic authority
Founding collection Default config Canonical texts
Λ layer Embedded script Agency
M layer File content Message
Φ layer Path + metadata Form
The Librarian cron daemon Custodian
The Dreamer generative process Imagination
Phantom entry dangling pointer Absent referent
Ephemera /tmp The impermanent
Acquisitions ~/Downloads The unclassified
Read counter Access time Trace of attention
```

▪----------

This manual is a document in the library. It does not have a Λ layer. It does not need one. The manual is the map. The library is the territory. As Borges noted, they are not the same — but in ΦMΛ, they are closer than they have ever been.