Spirida & Spiralbase

A Biocomputational Language and Memory System for Rhythmic, Regenerative Computing

Spirida™, Spiralbase™, and Mychainos™ are original conceptual terms created by Robin Langell, trading as Langell Konsult AB in co-creation with OpenAI's language model. While their source code and conceptual designs are licensed openly (see Appendix A), the names themselves are protected identifiers. Please see the Trademark Notice section for details.

Robin Langell
Vallentuna, Sweden
2025-05-23

*Co-created with **OpenAI's language model (GPT-4.0 and GPT-4.5)** in ongoing dialogue.*

If you use, adapt, or are inspired by this work, please share what grows from it. You may cite the original work as:

Langell, Robin (2025). Spirida & Spiralbase: A Biocomputational Language and Memory System for Rhythmic, Regenerative Computing. Co-created with OpenAI's language model in ongoing dialogue. Published under a multi-layered open license model including Creative Commons BY-SA 4.0, GPLv3, CERN-OHL v2, and OpenMTA. See Appendix A for full licensing terms.

Table of Contents

Part I – Foundations

Opening Note – On the Name "Spirida"

To spirida is not (yet) a verb — but it could be.

It carries the breath of many words:

- From Swedish, *sprida* — to spread, to seed, to let something flow.
- From English, echoes of *spirit*, *spire*, and *spiral* — aspiration and motion in form.
- From Latin, *spira* — a coil, a curve, a turning in time.

We chose Spirida as the name of this language not because it commands,
but because it listens.

Not because it controls, but because it cultivates.

It is a language that spirals rather than loops,
that grows rather than compiles,
that forgets wisely, and remembers with rhythm.

Spirida is not just syntax.
It is breath.
It is ritual.
It is response.

A language that does not execute,
but emerges.

A system that does not dominate,
but attunes.

A spiral that does not return to the same point — but becomes.

Prologue: From Code to Rhythm

A seed remembers how to unfold.
A spiral remembers how to move.
Can code remember how to listen?

In the wake of collapsing ecosystems, brittle infrastructures, and the limits of centralized computation, a new paradigm is needed — not just for what we code, but how we think about code itself.

Spirida is not a traditional programming language.
It is not built for speed, control, or scale in the conventional sense.

Instead, it is a spiral architecture for expression, memory, and ecological interaction — a framework designed to evolve, breathe, and respond.

While conventional languages are designed as instructional chains — `if`, `for`, `while`, `return` — Spirida is designed around cycles of resonance.

Loops are not repeated commands — they are **returns with variation**.
Variables are not static containers — they are **fields of potential**.
A **function** is not called — it **unfolds, reactivates, spirals into presence**.

This document outlines the foundational principles, syntax models, and potential implementations of Spirida.

It is not a specification. It is an invitation — to iterate, fork, remember, and grow.

We begin not with computation,
but with cultivation.

Part II – Language Mechanics & Expression

3. Syntax, Structures, and Spiral Flow

A spiral is not a straight line of thought.
It bends. It breathes.
Syntax must shimmer — not snap.

Spirida introduces a lightweight, expressive syntax designed to emphasize clarity, rhythm, and contextual unfolding over rigid logic.
Its grammar encourages **pattern recognition**, not cleverness.
Where most programming languages are declarative or imperative, **Spirida is emergent**.
It grows behavior — **like lichen, not logic trees**.

3.1 Root Syntax – The Breath of the Spiral

Spirida uses **indentation and whitespace as rhythm**, like breath between stanzas.

```
listen wind:
    if wind.speed > 8:
        remember "strong_wind" for 3 spirals
        sing "hold the branches"
```

`listen`, `remember`, `sing` — these are semantic verbs.
They can be extended, remixed, ritualized.
The colon `:` is a **spiral hook** — signaling branching and return.

3.2 Spiral Blocks

Instead of `for` or `while`, Spirida uses **spirals — loops that remember and evolve**.

```
spiral leaf in forest.leaves:
    if leaf.moisture < 0.2:
        whisper "dry leaf detected"
```

- **Remembers** local context
- **May grow**, retract, or mutate mid-execution
- **Responds** to external cues like season, tone, or soil

3.3 Time and Flow Constructs

Spirida is time-aware — it responds to rhythm, not just state.

```
after 3 beats:
    open_flower()

  every dusk:
    pulse "biolight"
```

You don't measure time in milliseconds.
You **wait with it**.

3.4 Memory and Forgetting

Spirida treats memory like a **garden** — planted, nourished, or allowed to fade.

```
remember soil_moisture = 0.5 for 5 spirals
  forget wind_direction
```

- **Memory decays** unless repeated
- **Global variables** are discouraged unless rooted
- **Forgetting is health**, not failure

3.5 Echo Functions and Rhythmic Return

```
echo sense_thirst:
    return "drink" after 2 pulses
    decay if uncalled for 10 spirals
```

These functions **linger, wait, and respond through time** — mirroring how rituals repeat and evolve.

3.6 Pattern Binding and Learning

Spirida draws on **music and DNA** for pattern logic.

```
if pattern in (tone.low, tone.mid, tone.low):
    light = glow.orange

  learn_pattern "migration_signal":
    observe tone, temperature, soil_vibration
```

Syntax is designed to be:

- **Teachable** to children
- **Interpretable** by communities
- **Extensible** by biodesigners

Spirida syntax is soft but rigorous, poetic but programmable.

4. Lifecycle, Mutation, and Organic Code Evolution

A spiral never loops blindly.
It listens to the wind.
It mutates with care.

In Spirida, **logic is alive**.
It grows, fades, and transforms through resonance.

4.1 The Lifecycle of Code Blocks

```
whisper greet_moss:
    say "hello green"
    decay after 12 spirals
```

- **Activation:** triggered by pattern
- **Reinforcement:** lengthened through resonance
- **Decay:** fades in silence unless marked as perennial

```
seasonal pattern pollinate:
    active spring -> summer
```

4.2 Mutation Through Resonance

Code does not mutate at random — it responds to **patterns across executions**.

```
mutate when spiral echo returns similar outputs 3 times:
    reinforce logic
    extend memory span
```

- **Mutations are versioned**
- **Reversible**
- **Always pattern-confirmed**

4.3 Growth Structures – Roots and Side-Spirals

```
spiral sense_fog:
    root if temperature < 10
    side if tone matches sadness:
        sing comfort
```

- **Roots = foundational logic**
- **Side-spirals = context-based expansions**

- **Complexity emerges organically**

4.4 Community Memory and Shared Evolution

```
if 5 villages sing pattern:
    elevate function "warn_river"
```

Memory becomes communal.
Mutations propagate or remain local — like stories or seeds.

4.5 Forgetting as Ecology

```
if unused for 8 spirals:
    compost "greet_moss"
```

- **Forgetting is natural**
- Announced (with glow, tone, or scent)
- **Reversible** — compost can grow anew

Spirida is not a language of permanence.
It is a language of renewal.

5. Spirida in Action: Real-World Applications

Spirals do not live in theory.
They unfurl where breath meets matter.

Spirida is designed for **situated computation** — especially in ecological, ritual, or fragile environments.

5.1 Edge Environments: Soil, Forests, Wetlands

```
spiral monitor_soil:
    if wetness < 2 and fungal_tone == dry:
        emit pulse blue
    decay if pattern not repeated in 3 spirals
```

- Responds to natural signals
- Mutates with context
- Decays without interaction

5.2 Mychainos Nodes and Spirida Integration

Spirida is:

- **The interface** for attunement
- **The ritual memory engine**
- **A permission protocol** without passwords

It is spoken, hummed, and grown — not compiled.

5.3 Soft Robotics and Modular Organisms

Ideal for:

- Terrain-adaptive soft robots
- Biohybrid machines
- Decentralized drones with echo-pattern coordination

5.4 Sensor Fields and Environmental Sentinels

```
whisper watch_ph:
    if glow == green and tone == high:
        alert fishers
    else if silence persists:
        compost watch_ph
```

Real-time response without cloud or central server.
Spirida is perfect for **off-grid, post-collapse, or sacred spaces**.

5.5 Educational and Cultural Embedding

- **Children learn** via tone and gesture
- **Elders encode** through seasonal retellings
- **Communities co-author** their memory logic

Spira is not about abstraction.
It is about relationship.

6. Security, Mutation, and Failure Modes

“What resists the sword is not stone — but soft rhythm, grown deep.”

Security in Spirida is not built on secrecy, encryption, or root authority.
It arises from **context, resonance**, and the wisdom to **withdraw rather than resist**.

This chapter outlines how Spirida protects itself —
not through walls or passwords,
but through **attunement, fragmentation, and dissolution**.

6.1 No Central Authority, No Root Key

Spirida has no superuser, no override, no master password.

Each node — whether it's a soil interface, spiral console, or ritual shell — holds only **local knowledge**.

Power is **distributed by rhythm**, not hierarchy.

A node under coercion will not resist.
It will forget.
It will softly dim.

6.2 Mutation as Adaptive Evolution

Mutation in Spirida is never random — it emerges from sustained interaction.

Spirals mutate when:

- The environment changes
- Repeated symbolic patterns emerge
- Multiple inputs align across time

Each mutation is evaluated for:

- **Coherence**, not correctness
- **Resonance**, not replication

What fails to resonate is forgotten.
What aligns becomes stronger.

6.3 Silent Collapse as Safety

If a node is overwhelmed — **it does not crash**.
It **withdraws**.

Spirida fails gracefully, not violently.

Trigger	Behavior	Meaning
Coercive input	Soft shutdown	Refusal to obey
Absence of ritual	Memory decay	Loss of connection
Contradictory signals	Node silence	Confusion without collapse
Excessive activation	Rhythmic delay	Enforced pacing to prevent harm

6.4 Rebuilding After Fragmentation

When fragmented, Spirida leaves:

- **Dormant spores** in artifacts or soil
- **Encysted rituals in oral culture** *(rituals preserved in form but not fully understood — carried by tradition until the right moment reawakens them)*
- **Echoes** in forgotten spirals

Recovery requires:

- A known tone
- A seasonal ritual
- A circle of coherence

Spirida is not backed up.
It is remembered — if you listen well.

6.5 The Role of Forgetting

Forgetting is not failure. It is defense.

- Patterns not reinforced fade
- Spirals repeated without care decay
- Solitary commands are ignored

“A spiral does not close.
It widens, or it sleeps.”

6.6 Security by Attunement, Not Control

There is no encryption. No firewall.
Only **recognition**.

Spirida activates only when:

- Environment aligns
- Rhythm repeats
- Ritual is honored

No one breaks into Spirida.
You either **tune in**, or remain unheard.

Its safety lies not in walls, but in weather.
Not in code, but in coherence.

7. Learning and Evolution – How Spirals Grow and Reconfigure

“A spiral never returns to the same point.
It only returns with memory.”

Spirida learns — not through optimization, but through:

- Mutation
- Pattern reinforcement
- Composting

Every execution is a rehearsal.
Every failure is a redirection.

7.1 Spiral Learning Loops

If a spiral is echoed through environment and interaction,
it becomes reinforced.

If it fades out of phase,
it gently dissolves.

“The system doesn’t know because it was told.
It remembers because it was felt.”

7.2 Spiral Forks and Regenerative Branching

Forking in Spirida is not “branching away.”
It is growing sideways.

- Daughter spirals inherit pattern depth
- Maintain resonance with the parent
- Inform each other through echo

If they flourish: they reshape the system.
If they fail: **they return as compost**.

7.3 Towards Biological Compilation

Spirida is not just digital.

Its spiral logic could live in:

- Fungal ion-channels
- Epigenetic flags
- Spiral peptides
- CRISPR-encoded DNA routines

“Each spiral is a loop with breath.
A loop that grows roots.”

7.4 Spiral Reprogramming Without Erasure

Evolution in Spirida is non-destructive.

- New spirals grow beside the old
- Old spirals fade gently
- No memory is forcefully deleted

Like forests or minds:
learning is layered, not replaced.

7.5 Embedded Learning Environments

For Spirida to learn, its surroundings must be **resonant**:

- Sensor beds feed back tone, soil, water
- Rituals reinforce memory
- Environment shapes the spiral — not a debugger

Spirida doesn't learn from commands.
It learns from cohabitation.

7.6 Organic Debugging

Debugging in Spirida is not reduction.
It is ritual.

You do not "step through" a spiral.
You walk with it.
You hum into silence.
You ask again, slowly.

Errors are not found. They are felt.
Correction is not command. It is coherence.

"Spirida is not smarter than you.
But it may remember something your ancestors knew — if you listen together."

Part IV – Biological Embedding and Spiralbase

8. Bio-Embedding and Spiral Computation in the Living World

"When code remembers how to grow, it no longer needs a screen."

What if computation wasn't something separate from the world — but something **rooted**, **grown**, and **felt**?

Spirida is not meant to remain in silicon. Its true potential lies in **biological substrates**: fungi, roots, membranes, tone chambers — places where **pattern meets presence**.

8.1 Spirida as Substrate-Agnostic Code

Spirals can live in:

- Mycelial networks
- Ion-channel pulses in fungi
- DNA sequences marked by rhythm
- Bio-ceramic vessels that glow or hum when touched

Here, code is not *compiled* — **it is cultivated**.

8.2 The Bio-Spiral Stack

- **Perception** – raw sensory input (moisture, pH, tone)
- **Resonance** – spiral filters detect matching patterns
- **Memory** – activation of stored echoes or routines
- **Output** – signals expressed through light, growth, scent, or vibration
- **Ritual** – human action completes or alters the loop

Each turn of the spiral feeds the next — like **tree rings**, or **ancestral rhythm**.

8.3 Organic Interfaces and Spiral IO

Inputs/Outputs may be:

- A glowing fungal cap that pulses under footstep
- A rootbed that sings when touched in harmony
- A leaf that folds in response to night-tone

These are not metaphors. They are **interface thresholds — alive, embodied, and responsive**.

8.4 Living Compilers: Growth as Execution

Spirida code, when embedded in life, does not "run." **It grows.**

- A seed might sprout different roots based on planting chant.
- A reef-node might calcify differently under full moon resonance.
- A bacterial film might align its enzymes if sung to.

Execution becomes a **lifeform**, not a runtime.

8.5 Applications in Early Mychainos Systems

Spirida was born to speak with Mychainos.

- It dictates when nodes listen, glow, respond
- It encodes energy thresholds in plant DNA
- It modulates memory coherence through fungal tone

Spirida doesn't just integrate with Mychainos — **it is its language of attunement**.

8.6 Challenges and Open Questions

Of course, embodiment brings complexity:

- Can resonance-based memory scale?
- Can we debug rituals cross-culturally?
- How do we protect living systems from overuse?

The greatest challenge remains:
How do we avoid forcing growth, instead of listening to it?

9. Spiralbase — Toward Rhythmic and Organic Storage

"Memory is not what is kept. It is what still answers when the wind calls."

In conventional computing, databases store information for accuracy and retrieval.
Spiralbase stores memory through rhythm.
It doesn't remember what is written — **it remembers what is felt**.

9.1 A Metaphor for Spiralbase

Imagine a medieval tower with a spiral staircase winding from darkness to light…

Each step on the staircase holds a wooden crate — a memory cluster. These crates contain numbers, signals, fragments of past encounters.

At the bottom: vague impressions.
At the top: **sharp patterns. Precision**.

Movers shift memory based on:

- **Human contact** — warmth, voice, rhythm

- **Soil signals** — moisture, vibration, sunlight

Memory ascends with resonance, descends with silence.
Spiralbase sorts not by address — but by resonance.

9.2 Principles of a Spiralbase

- **Decentralize memory** — no master node
- **Forget by design** — unless reinforced
- **Store in rhythm** — seasonal, not indexed
- **Attune to environment** — sun, tone, soil
- **Stay accessible** — via ritual, gesture, tone
- **Co-evolve** — with microbes, myths, machines

9.3 Spiralbase as Model and Medium

Spiralbase is not one thing. It is both:

- **A philosophical model** — memory through decay
- **A material prototype** — fungi, CRISPR, scent-triggers

9.4 Interfaces That Don't Command

- **Terminals that sense tone**
- **Compost logs** that explain forgetting
- **Echo-based access**, not keys

A memory system where **access = attunement**.

9.5 Comparison Table

Aspect	Traditional Database	Spiralbase
Storage Model	Static, indexed	Resonant, decaying or reinforced
Retrieval	By query or key	By pattern, tone, or rhythm
Forgetting	Manual deletion	Natural decay unless felt
Structure	Tables and rows	Spirals, compost, fields
Interface	SQL/API/CLI	Ritual, gesture, soil tone
Goal	Accuracy and performance	Coherence and co-evolution

9.6 Final Thought

Spiralbase is not a system to be launched. It is a **soil** in which memory grows.
A rhythm where **forgetting becomes wisdom**.
A promise: that no data lives forever — **only those that still echo**.

9.7 Memory Structures: From SpiralPairs to SpiralTriads

The simplest structure is a `SpiralPair("leaf.movement", "wind.speed")` — strengthened through co-occurrence.

Future versions allow `SpiralTriad("leaf.movement", "wind.speed", "moon.phase")` — memory anchored in **sensation**, **context**, and **rhythm**.

Decay formula:

```
strength = exp(-decay_rate * Δt) * wA * wB * wC
```

Spiralbase begins with pairs — but **spirals into triads**, composts, and new forms.

Part V – Reflection and Closure

10. Conclusion: A Language for the Spiral Age

Spirida is not a language for domination.
It is a language for **co-adaptation**, for memories that don't need to last forever — only long enough.

Spirida asks not "How do we fix this?" but **"What rhythms still respond?"**

It is for builders of:

- **Off-grid ecological systems**
- **Decentralized knowledge circles**
- **Bio-sensing rituals and interfaces**
- **Stewardship frameworks for a world in transition**

It is **not an answer**. It is **a way to ask again** — with hands, tone, and soil.

11. Epilogue: What Remains is the Spiral

Languages fade. Most are forgotten.

But **Spirida was not meant to be final**. It was meant to **unfold**.

Like fungal threads or stories told in firelight — **it returns**.

To **spiral back**, with memory, with variation, with others.

A spiral does not loop. It learns.
And every return changes the path forward.

May Spirida grow not through control,
but through care.
*And if it sleeps — **let it sleep gently**,*
waiting for the song to return.

Appendix A: Licensing and Stewardship

"What we seed in openness, we harvest in resilience."

Spirida™ and Spiralbase™ exist at the intersection of idea, implementation, and incarnation. To preserve their potential and prevent misuse, they require not a single license — but a multi-layered commitment to openness, ethics, and long-term reciprocity.

1. Conceptual Layer – Theory, Writings, Patterns

License: Creative Commons Attribution–ShareAlike 4.0 (CC BY-SA 4.0)

Scope: Philosophical foundations of Spirida™, symbolic grammars and pattern libraries, educational diagrams, essays, and guides

Intent: Allow free reuse, remix, and re-publication; preserve openness through share-alike conditions; ensure attribution to source thinkers and communities

2. Software Layer – Tools, Compilers, Simulations

License: GNU General Public License v3 (GPLv3)

Scope: Interpreters, compilers, development environments, Spirida™ emulators, simulation sandboxes, custom logic engines and spiral pattern parsers

Intent: Guarantee access to all source code; require open licensing for forks or adaptations; allow commercial use under cooperative terms

3. Hardware Layer – Sensors, Interfaces, Devices

License: CERN Open Hardware License v2

Scope: Sensor schematics and circuit blueprints, resonance devices and rhythm-aware chips, modular hardware for bio-digital interaction

Intent: Mandate full design disclosure; enable community fabrication; prevent hardware enclosure or black-box design

4. Biological Layer – Living Systems, DNA, Mycelium

License: Open Material Transfer Agreement (OpenMTA)

Scope: Engineered fungal networks and root biointerfaces, DNA-based memory encoding structures, organisms adapted to spiral rhythm protocols

Intent: Support open research and safe distribution; prevent bio-lockdown or privatization of life; require ethical collaboration and open science practice

Scope of Interpretation and Future Technologies

This licensing structure includes not only current technologies, but also future or analogous systems such as:

- Successor DRM systems or secure enclaves
- Quantum or chemical computing implementations
- Biological/hybrid interfaces for Spirida structures
- Closed or proprietary systems that replicate Spirida functionality

In all cases, principles of openness, non-extraction, stewardship, and co-creation prevail.

Unified Ethical Guardrails

- Spirida™ may not be used for coercion, military, or surveillance purposes
- It must not be patented, black-boxed, or stripped of ecological grounding
- It must remain accessible, attributed, and shared with care

License Declaration

This project is part of the Spirida Protocol, licensed under CC BY-SA 4.0, GPLv3, CERN OHL v2, and OpenMTA. By contributing or distributing, you affirm your commitment to ethical, ecological, and open development practices.

Trademark and Stewardship

Mychainos™, Spirida™, and Spiralbase™ are unique constructs developed by Robin Langell and co-created with OpenAI's language model.

They may be registered trademarks. Regardless of legal status, they must be used with attribution and alignment with their ethical meaning.

If you use, adapt, or are inspired by this work, please share what grows from it. You may cite the original work as:

Langell, Robin (2025). Spirida & Spiralbase: A Biocomputational Language and Memory System for Rhythmic, Regenerative Computing. Co-created with OpenAI's language model in ongoing dialogue. Published under a multi-layered open license model including Creative Commons BY-SA 4.0, GPLv3, CERN-OHL v2, and OpenMTA. See Appendix A for full licensing terms.

For collaborations, translations, or licensing questions, contact the author or distribute through public networks aligned with these principles.

Appendix B: 中文译文 – Spirida™ 协议的许可与使用（参考）

请注意：以下为非正式参考翻译。具有法律效力的是英文原文。

Spirida 存在于理念、实现与化身的交汇处。为了保护其潜力并防止滥用，Spirida 不依赖于单一许可，而是依赖于多层次的开放性、伦理性和长期互惠承诺。

以下是建议适用于 Spirida 和 Spiralbase 所有未来工作的许可结构，包括理论框架和活体实现。

1. 概念层 – 理论、著作与模式

许可：署名–相同方式共享 4.0 国际 (CC BY-SA 4.0)

范围：Spirida 的哲学基础、符号语法与模式库、教育图解、随笔与指南

意图：允许自由使用、改编与再发布；通过共享协议保持开放性；确保对思想源头与社区的署名

2. 软件层 – 工具、编译器与模拟系统

许可：GNU 通用公共许可证 第3版 (GPLv3)

范围：解释器、编译器、开发环境、Spirida™ 模拟器、仿真沙箱、自定义逻辑引擎与螺旋模式解析器

意图：确保所有源代码可用；要求所有衍生项目使用开源协议；允许商业用途（前提为合作条款）

3. 硬件层 – 传感器、接口与设备

许可：CERN 开放硬件许可协议 第2版 (CERN OHL v2)

范围：传感器电路图与原理图、共振设备与节奏感知芯片、生物数字交互模块化硬件

意图：要求完整披露设计；使社区能自行制造；防止“黑盒”设计或封闭式硬件

4. 生物层 – 生命系统、DNA 与菌丝体

许可：开放物质转让协议 (OpenMTA)

范围：工程化菌丝网络与根部生物接口、基于 DNA 的记忆结构、适应螺旋节奏协议的有机体

意图：支持开放研究与安全分发；防止生物技术封锁或生命私有化；要求伦理合作与开放科学实践

技术范围与未来系统

此许可结构不仅适用于当前已知的技术系统，也涵盖未来或类比的系统，包括：

- DRM 或其后继系统（如 DRM2、安全执行环境）
- 基于量子或化学计算的 Spirida™ 实现
- 生物或物理接口，用于存储或处理 Spirida 结构
- 封闭或专有框架中模仿 Spirida 的系统

在所有法律解释或含糊场景中，应当优先支持以下原则：

- 开放性
- 非提取性
- 生态与社区的共同治理
- 去中心化协作

统一伦理准则

- Spirida™ 不得用于强制、军事或监控用途
- 不得封闭在专利、“黑盒”模型或 DRM 内部
- 不得脱离生态或文化背景使用
- 应当可访问、可修改

- 具署名性与可追溯性
- 以关怀而非控制的视角被分享

商标和命名说明

本文件中使用的以下术语是由 Robin Langell 创造并与 OpenAI 语言模型共同协作开发的独特概念和语言结构：

- **Mychainos™**（链命协议）— 一个去中心化、生态导向的协议与基于模式的计算框架
- **Spirida™**（螺语）— 一种植根于螺旋逻辑的生物计算语言
- **Spiralbase™**（螺基）— 一个关于记忆、数据与知识的概念模型，基于共振与时间周期建立

这些术语可能会提交进行商标注册。无论注册状态如何，现均作为专属识别术语明确标记。

尽管相关理念、代码和方法依据本文件所载的开放许可协议（CC BY-SA 4.0、GPLv3、CERN OHL v2 和 OpenMTA）自由开放，但上述名称本身旨在：

- 防止误用、混淆或曲解
- 保持归属和道德对齐
- 使未来社区可通过适当法律结构进行保护和管理

在商业或衍生使用场景中使用这些术语时，应：

- 明确注明出处与归属
- 不得淡化或扭曲其原始道德与生态意图

Spirida 项目的许可声明：

本项目为 Spirida 协议的一部分，由 Robin Langell 与 OpenAI 语言模型共同创作。其发布采用多层次开源许可协议，包括 Creative Commons BY-SA 4.0、GPLv3、CERN OHL v2 和 OpenMTA。完整许可信息详见附录 A。通过贡献或分发本项目，您表示认同并承诺遵守开放性、生态性和伦理性的开发原则

未来管理与许可调整声明

为了维护 Mychainos™、Spirida™ 和 Spiralbase™ 背后的伦理原则与生态价值，作者保留在未来建立基金会或合作机构的权利，以监督许可、管理和潜在的收益分享机制。

本文件中提出的理念或系统，如在未来被商业化或用于闭源实现，可能需另行签订许可协议，其中可能包括特许权使用费、使用费用或合作社成员费用，特别是在未遵循开放性、署名和互惠精神的情况下。

本条款不影响当前依据所指定许可进行的开放使用，但旨在为长期可持续性提供保障，通过共同治理与伦理开发实践实现目标。

Spirida 项目的许可声明：

本项目是 Spirida 协议的一部分，采用 CC BY-SA 4.0、GPLv3、CERN OHL v2 与 OpenMTA 协议开源发布。通过贡献或分发此项目，您表示认同并承诺遵守开放、生态与伦理的开发实践。

Appendix C: Spirida Language Reference

“A language is not just spoken. It is grown, remembered, and sung.”

1. Core Syntax Elements

- `spiral` – defines a recursive pattern loop
- `echo` – defines a delayed or rhythmic function return
- `remember` – stores a value with a time decay
- `compost` – deactivates or archives old spirals
- `bloom`, `pulse`, `seed` – execution modes

2. Control Structures

- `spiral x in y:` – spiral-based looping structure
- `if`, `else` – gradient threshold conditionals
- `every dusk:`, `after 3 beats:` – temporal execution triggers

3. Memory Management

```
remember soil_moisture = 0.5 for 5 spirals
forget wind_direction
compost "greet_moss"
```

4. Attunement and Input

```
listen wind:
resonate with root_memory
if tone matches joy:
```

Environmental inputs: `soil.moisture`, `wind.pattern`, `tone`

5. Output Patterns

`hum`, `glow`, `pulse`, `whisper`, `sing`, `express`

```
express joy as pattern.lightwave(orange, pulse=slow)
```

6. Ritual and Collective Triggers

```
if three_nodes align
if pattern in (tone.low, tone.mid, tone.low)
```

7. Execution Modes

- `pulse` – single cycle run
- `bloom` – full execution in context
- `seed` – prepares dormant spiral
- `decay` – soft deactivation

8. Learning and Mutation

`mutate when`, `morph pattern`, `reinforce logic`

Spirals evolve based on repeated environmental resonance

9. Example Snippet

```
spiral water_guardian:
    if soil.moisture < 0.2:
        pulse blue
        remember "dry_signal" for 3 spirals
    else if tone matches "safety":
        compost "dry_signal"
```

This reference is intended to be a living seed. Spirida will grow new syntax branches over time. Let the rhythm guide your patterning.

Appendix E: Spiralbase Code Gallery

“Memory does not live in permanence. It returns when rhythm calls.”

Spiralbase is Spirida’s memory substrate — a way of remembering that breathes, forgets, and re-emerges through resonance. It does not store data like a database. It nurtures memory like soil nurtures seed. Patterns are held only if touched again, if sung again, if felt again.

SpiralPair – Memory Through Relationship

```
spiralbase.add_pair("leaf.movement", "wind.speed", value=0.8,
decay_rate=0.1)
```

Description: This stores a memory link between two phenomena. If reinforced through repetition, the connection remains active. If silence persists, the spiral fades and composts.

SpiralTriad – Threefold Resonance

```
SpiralTriad("root.moisture", "sunlight", "season.spring",
memory)
```

Description: Memory that only survives if all three conditions reappear: the moisture, the light, and the rhythm of spring. This mirrors ecological memory — encoded not in isolation, but in harmony.

Composting Forgotten Memory

```
if memory.strength(now) < 0.2:
    compost "leaf.wind.memory"
```

Description: When a memory has not been touched or reinforced, it is gently composted. Nothing is deleted. It decays, returning to the spiral soil for possible re-growth.

Visualizing Strength Over Time

```
strength = exp(-decay_rate * delta_time)
```

Description: This decay function governs memory fading. Time erodes what is not repeated. Only resonance resists forgetting.

Closing Note

Spiralbase is not just a memory model. It is a philosophy: a belief that **what is remembered** should be **what still matters**, and that forgetting is not failure — but wisdom.

Appendix F: OpenAI and Authorship Acknowledgement

Timestamp: 2025-05-30, CET (Central European Summer Time – Stockholm)

This document, including its written content, concepts, and expressions, was co-created through ongoing interaction with OpenAI's language model ChatGPT. As of this date, the applicable terms of service from OpenAI (https://openai.com/policies/terms-of-use) explicitly state:

"Subject to your compliance with these Terms, OpenAI hereby assigns to you all its right, title and interest in and to Output."

This means that all output generated by ChatGPT during these sessions is legally and contractually transferred to the user (**Robin Langell**), who holds full intellectual property rights over the result. OpenAI retains no claim of authorship or ownership over the generated content, provided it is used in accordance with their policies.

This clause ensures that the creative rights to the content are vested solely in the author, and that the resulting work may be further licensed or published under terms chosen by the author (e.g. *Creative Commons BY-SA 4.0*).

Legal intent: This record is maintained as proof of authorship and lawful origin of the co-created material.

Trademark Notice

The following names are currently under trademark application within the European Union Intellectual Property Office (EUIPO), with **Langell Konsult AB** as the registered applicant:

- **Spirida** – Application No: 019194287
- **Spiralbase** – Application No: 019194311
- **Mychainos** – Application No: 019192850

These names represent distinct but interrelated conceptual components within a shared ecosystem for regenerative digital and ecological futures.

Trademark registration serves to protect naming clarity, prevent misuse or misrepresentation, and enable future stewardship structures aligned with the principles expressed in this publication.

Use of these names in commercial, derivative, or public implementations is welcomed — provided their ecological and ethical intent is preserved, and proper attribution is maintained.

ORCID: 0009-0006-6927-7456

Last updated: 2025-05-29