NeuroShellOS: An AI-Integrated Operating System – Part 1

"The OS that thinks with you"

A Sample Application of AI-Native Operating System Concepts

Author: Muhammed Shafin P

Executive Summary

What is NeuroShellOS?

NeuroShellOS is a conceptual operating system design that explores how artificial intelligence can be deeply integrated into the core architecture of a computer system. Unlike traditional operating systems where AI is an optional add-on or separate application, this concept proposes making AI a fundamental part of how users interact with their computers.

Key Benefits

Natural Language Interaction: Users could communicate with their computer using everyday language instead of memorizing complex commands or navigating through multiple menus.

Contextual Intelligence: The system would understand what you're currently working on and provide relevant assistance without being asked.

Privacy-First Design: All AI processing happens locally on your device, ensuring your data and activities remain private.

AI-Assisted Setup: The system simplifies complex Linux environment setups, allowing developers and hackers to skip manual configuration—AI handles it for them.

Offline Functionality: The system works completely without internet connection, making it reliable and secure.

Lightweight and Efficient: Even with built-in AI, the system leverages Linux's performance and adopts an on-demand execution strategy to reduce background resource usage. It uses significantly less RAM than traditional OSes like Windows, offering better speed and control.

Customizable Experience: Users maintain complete control over AI behavior and can disable features they don't want.

Open Source Philosophy: The concept encourages community development and transparency in AI integration.

This Document's Purpose

This document represents **Part 1(also Sample 1) of a practical exploration of AI-integrated operating systems. NeuroShellOS** serves as a sample application demonstrating how the base principles of AI-native computing can be implemented in real-world scenarios. This is one possible interpretation of how AI can be deeply integrated into operating system architecture while maintaining user privacy and control.

Introduction

The Vision Behind NeuroShellOS

NeuroShellOS is built on the belief that an operating system should be intelligent, adaptable, and accessible. While traditional systems often require technical expertise and manual configuration, NeuroShellOS bridges the gap by offering a smarter environment that understands user intent and simplifies complex tasks.

By leveraging the power of Linux and integrating AI at its core, NeuroShellOS promotes digital literacy—empowering users to interact with their system more naturally, learn by doing, and explore advanced computing without barriers.

This isn't just another OS; it's the foundation of a self-evolving, intelligent platform designed to become the ultimate interface between humans and machines.

Why AI-Native Operating Systems?

In the past, operating systems were silent engines—powerful, but passive. They waited for commands, offered little guidance, and left users to bridge the gap between human thought and machine logic. But the world has changed. We no longer need machines that simply obey—we need systems that understand.

AI-native operating systems represent a new chapter in human-computer interaction. They are not just tools; they are intelligent companions—systems that learn, adapt, and respond. They serve as co-pilots for developers, guides for newcomers, and partners for creators, thinkers, and builders.

NeuroShellOS is built to be just that: a thinking environment. It reduces friction, automates the complex, and illuminates possibilities the user might not even know existed. It is designed to elevate digital literacy by removing technical barriers and making advanced computing feel as natural as speaking or writing.

AI-native OSes aren't just about efficiency—they’re about harmony between human and machine. A shift from command to collaboration. A future where your OS doesn’t just run your system—it works *with you*.

Our Approach to Privacy and Control

NeuroShellOS is built on the belief that intelligence should never compromise privacy. While modern systems often trade convenience for control, we choose a different path—one where the user decides everything.

All AI processing runs locally by default, ensuring your data and activity remain fully private. Nothing is sent or shared without your explicit permission. If the system offers features to improve itself—such as optional data contribution to expand AI capabilities—it is always up to you. You choose what to share, when to share, and whether to turn those features on or off entirely.

From AI behavior to system-level controls, every function in NeuroShellOS is transparent, customizable, and owned by you. Because real intelligence respects freedom—and puts users in command.

Core Principles

AI as System Infrastructure

In traditional operating systems, intelligence is layered on top—detached, reactive, and often optional. But NeuroShellOS reimagines this completely. Here, AI is not an accessory; it is the very foundation upon which the system breathes, moves, and evolves.

This is not about running a chatbot or launching a voice assistant. It's about an operating system that understands its user, responds to context, and shapes itself in real time. The AI is woven deep into the system's fabric, quietly orchestrating decisions, optimizing performance, and offering meaningful insights without being told.

Imagine an OS that doesn't wait for you to troubleshoot—it senses the abnormality before it becomes a problem. One that configures your development environment as you begin typing, anticipates the tools you'll need, and fades into the background when you don't.

In NeuroShellOS, AI is not a tool. It is the **pulse** of the system. Thoughtful, quiet, and always present—transforming the machine from a passive servant into an active collaborator.

User Privacy and Control

In an age where data is currency and surveillance is subtle, NeuroShellOS takes a bold and necessary stance: **the system belongs to the user—entirely**.

There are no silent observers here. No hidden trackers whispering in the background. Every process, every AI response, every background decision is visible, accountable, and, most importantly—**optional**. Privacy is not a setting. It is the default.

Users are not merely allowed control—they are **empowered** with it. From turning off telemetry to choosing whether AI features can access certain files, every choice rests in the user's hands. Even data sharing, when used to expand the intelligence of the system, is always **opt-in**, never assumed.

NeuroShellOS believes that intelligence without consent is intrusion. That automation without control is confinement. Here, privacy is not sacrificed for progress—it **guides** it.

In this operating system, the user is not just a participant.
They are the sovereign.

Offline-First Design

In a world that increasingly demands constant connectivity, NeuroShellOS offers a rare and radical promise: **full intelligence, even in solitude**.

Where most systems degrade without the internet—dependent on cloud servers, external APIs, and remote verification—NeuroShellOS remains fully functional, fully aware, and fully yours. The intelligence doesn't vanish when the signal fades. It remains seated at the heart of the OS, ready to assist, analyze, and adapt—**no connection required**.

Yes, cloud AI integration is available—for those who want advanced capabilities or distributed support. But here, **offline is not the fallback. It is the default.** The system is designed to think locally, act immediately, and protect your data even in total isolation.

This is more than a technical feature—it's a philosophy.
An affirmation that autonomy matters. That your OS should serve you whether on the open web or in complete silence.

In NeuroShellOS, being disconnected is not a limitation.
It is a state of freedom—where your system remains intelligent, private, and empowered by design.

Open Source Philosophy

NeuroShellOS is not built behind closed doors. It is not guarded by proprietary walls or veiled by hidden code. Instead, it is shaped in the open—**by people, for people**.

This system stands on the shoulders of the free and the fearless: the global communities of developers, thinkers, and rebels who believe that software should be shared, understood, and improved by all. Every line of code in NeuroShellOS is open for inspection, adaptation, and contribution.

This isn't just transparency—it's trust. The ability to know how your system works, to bend it to your will, to take part in its evolution. Whether you're reading the source, rewriting a module, or simply choosing an AI model that aligns with your values, you are part of something larger.

NeuroShellOS is more than software.
It is a movement—a declaration that intelligence, privacy, and control should never be locked away.

Open source is not a feature.
It is the soul of the system.

Key Features Overview

Natural Language Interface

NeuroShellOS reimagines the relationship between user and system—not through commands and code alone, but through natural conversation. Instead of memorizing syntax or navigating deep menus, users can simply describe their intent, and the system interprets it with intelligence and accuracy. Whether installing a tool, configuring an environment, or querying system health, interaction becomes direct, intuitive, and seamless.

Central to this design is a next-generation interface layer, where the **GUI, shell, and AI are not distinct surfaces, but parts of one unified system**. The shell interprets logic, the GUI presents clarity, and the AI brings understanding—all blending into a fluid user experience. There is no transition between modes or layers; the system flows naturally based on how the user engages.

This natural interaction is powered by a **flexible and scalable AI architecture**, built to support multiple modes of intelligent execution depending on task complexity, performance needs, and available resources. The system supports:

- **LLM + Scripts**: where a large language model interprets the user's request and directly maps it to fast, modular scripts written in Python, C++, Rust, or other suitable languages. Ideal for tasks like automation, system configuration, or package installation.
- **LLM + ML + Scripts**: used when deeper context, behavioral learning, or usage history needs to inform decision-making—such as log diagnostics, resource optimization, or anomaly detection.
- **LLM + ML**: for real-time suggestions, adaptive UIs, or predictive operations that don't require direct system changes but offer smart guidance.
- **LLM-only**: used in lightweight, offline-first environments, where basic interpretation and command mapping are needed without additional components.
- **Scripts-only fallback**: for ultra-minimal environments, where users prefer direct manual scripting with no AI involvement at all.
- **Cloud-only**, where all AI reasoning occurs through an online model, with minimal local impact
- **LLM + Scripts + Cloud**, blending fast local actions with extended cloud-based context
- **LLM + ML + Scripts + Cloud**, offering full-system cognition, personalization, and automation

- **Scripts + Cloud,** Local scripts run, but cloud assists with some remote intelligence or data storage, no LLM locally.
- **ML + Cloud,** ML happens in local environment, without local LLM or scripts.
- **LLM + ML + Cloud,** LLM models run in the cloud, to save local resources, still using AI with behavioral learning but no local scripts.

These modules are **modular and on-demand**—lightweight ML models and scripts only load when required, ensuring that the system remains fast, responsive, and resource-conscious, even on lower-end hardware.

For instance, when a user downloads a project from GitHub, they no longer need to manually read the README, resolve dependencies, and debug installation. Instead, they can simply say, "Install this project with its dependencies," or "Run this in isolated developer mode." The system intelligently reads the structure, understands the context, and performs the required actions—with full transparency and optional user approval.

Importantly, the presence of AI does not restrict the user. NeuroShellOS is fully usable in **both AI-enhanced and manual modes**. Users can switch to direct shell access, run scripts manually, write code, or use traditional package managers. The AI is a co-pilot—not a gatekeeper—and its behavior can be enabled, disabled, tuned, or extended at any time.

What makes this interface even more powerful is its **portability**. With official extensions, the same natural language layer can be embedded into applications like **VSCode**, terminal emulators, and developer dashboards—allowing users to access NeuroShellOS intelligence even when working outside the core OS interface.

This is more than just a feature. It's a philosophy: interaction should be expressive, flexible, and never limited by technical friction. NeuroShellOS's natural language interface brings the user and system closer together—merging intelligence, clarity, and control into a single, seamless experience.

Smart System Assistant

At the heart of NeuroShellOS lies a deeply integrated, intelligent system assistant—designed not as a layer on top, but as a native component of the OS itself. This assistant is not a passive add-on; it is a responsive, observant, and adaptive part of the system's core. Its purpose is to reduce friction, boost productivity, and expand user capability without unnecessary intrusion.

This assistant can configure secure developer environments, automate installations, monitor performance, and respond to unusual system activity with precision. Whether debugging

code, setting up runtime environments, or managing updates, the assistant offers relevant, timely support—always in tune with user goals, and always respectful of their space.

What makes this assistant unique is how deeply it is woven into the visual and functional structure of the OS. **A dedicated GUI package has been created exclusively for NeuroShellOS**—not simply to make the system look modern, but to reimagine interaction around intelligence. This GUI is lightweight yet powerful, designed from the ground up for responsiveness, clarity, and adaptability. It is more than a desktop; it is a canvas for intelligent interaction.

The GUI supports **custom themes, modular tile layouts, dynamic dashboards**, and **real-time visual extensions** of assistant actions. Users can personalize their interface deeply—rearranging elements, applying curated styles, or even creating and sharing their own UI packages. For developers and designers, the system allows **complete GUI extensibility**, enabling the creation of custom control panels, skins, and assistant-integrated interfaces with minimal overhead. **All customization can be performed manually or with the support of the built-in AI**. When AI is enabled, the customization process becomes significantly simpler—suggesting layouts, adapting to usage patterns, and applying optimized designs based on user preferences. Whether handcrafted or guided by intelligence, the interface remains deeply flexible, giving users full control over how their environment looks and behaves.

The Smart Assistant lives inside this interface—not as a widget, but as a layer of awareness that informs and enhances every element. It helps users navigate, acts as a guide through settings, configures systems visually, and adapts the GUI based on activity or preferences. The result is not just a smart assistant—but a **smart interface**, where the environment itself evolves with the user.

Behind the scenes, the assistant operates through a **modular AI execution framework** that supports various intelligent configurations:

- **LLM + Scripts**: where a large language model interprets the user's request and directly maps it to fast, modular scripts written in Python, C++, Rust, or other suitable languages. Ideal for tasks like automation, system configuration, or package installation.
- **LLM + ML + Scripts**: used when deeper context, behavioral learning, or usage history needs to inform decision-making—such as log diagnostics, resource optimization, or anomaly detection.
- **LLM + ML**: for real-time suggestions, adaptive UIs, or predictive operations that don't require direct system changes but offer smart guidance.
- **LLM-only**: used in lightweight, offline-first environments, where basic interpretation and command mapping are needed without additional components.

- **Scripts-only fallback**: for ultra-minimal environments, where users prefer direct manual scripting with no AI involvement at all.
- **Cloud-only**, where all AI reasoning occurs through an online model, with minimal local impact
- **LLM + Scripts + Cloud**, blending fast local actions with extended cloud-based context
- **LLM + ML + Scripts + Cloud**, offering full-system cognition, personalization, and automation
- **Scripts + Cloud,** Local scripts run, but cloud assists with some remote intelligence or data storage, no LLM locally.
- **ML + Cloud,** ML happens in local environment, without local LLM or scripts.
- **LLM + ML + Cloud,** LLM models run in the cloud, to save local resources, still using AI with behavioral learning but no local scripts.

Each of these modes is **on-demand and configurable**. Lightweight scripts or ML models are triggered only when necessary, ensuring high performance even on modest hardware. By default, the assistant runs locally with minimal footprint, but when users **enable cloud connectivity**, it expands—gaining access to large-scale reasoning, persistent memory, and broader integrations.

Whether used for intelligent help, automation, customization, or creative expression, the assistant stays within bounds. It suggests, it assists—but it never overrides. And when paired with the user-defined GUI, it becomes an **adaptive visual partner**, reflecting not only the user's commands—but their style, habits, and preferred flow.

In NeuroShellOS, the Smart System Assistant is not an accessory. It is a silent co-creator—present across the system, visible through its evolving interface, and aligned with the user at every level of design, function, and decision-making.

User Privacy and Control

NeuroShellOS is built on the belief that intelligence should never come at the cost of autonomy. In a world where AI-driven systems often blur the boundaries between helpful and invasive, NeuroShellOS takes a clear, deliberate stance: the user is always in control.

Every intelligent feature—from the assistant to background models—is transparent, optional, and tunable. The system does not collect personal data unless explicitly permitted. Users can inspect, pause, or disable any component of the AI infrastructure at any time, without affecting the core functionality of the OS. By default, all key features run offline, and cloud connectivity is never assumed.

Users decide whether to allow system data to be used to improve AI behavior, or whether to opt out entirely. Even when data-sharing is enabled, the system makes it clear what information is being accessed, where it is processed, and for what purpose. There are no silent uploads, no hidden analytics, and no locked settings.

This philosophy of control extends into the system assistant, automation routines, and ML models. All logs, learning patterns, and smart behavior are **local-first** by design. The assistant does not retain session memory unless the user requests it. The user interface itself offers visibility into every action the AI takes, and every recommendation is delivered with reasoning—never as a black box.

Most importantly, if the AI ever initiates an action that is **irreversible**, critical to system integrity, or impacts security or personal data, it will **pause and request user confirmation**. This may come in the form of a prompt to **resubmit**, **verify intent**, or **review the implications** before proceeding. NeuroShellOS does not allow any AI-driven process to make permanent decisions without clear, informed user approval.

To give users layered control, **system settings are divided into two primary GUI modes**:

- A **Basic mode**, with streamlined and essential options for everyday use.
- An **Advanced mode**, which unlocks broader controls over AI behavior, privacy management, system optimization, execution layers, and feature tuning.

A **toggle switch** within the GUI allows seamless switching between Basic and Advanced modes. In Advanced mode, users can see and control deeper parts of the system. From here, they may also **activate Developer Mode**—a powerful configuration layer meant for advanced users.

Developer Mode includes highly detailed options such as real-time model tuning, assistant debugging tools, sandbox configuration, internal service management, and experimental settings. These controls may appear as additional GUI panels, CLI overlays, or hybrid tools—depending on module complexity. This mode can be safely **deactivated**, returning the user back to Advanced mode, and then optionally back to Basic via the same toggle mechanism.

When a user chooses to integrate **cloud-based models**, that decision is made explicitly. No automatic switch to cloud occurs. And even then, users can select exactly which features use the cloud, define what is allowed to leave the system, and toggle those permissions at any time.

NeuroShellOS treats **privacy not as a settings page, but as a right built into the system architecture**. From initial setup to daily use, the system communicates with clarity, respects user agency, and ensures that every action the AI takes is both visible and reversible—unless the user agrees otherwise.

In an ecosystem where AI is everywhere, NeuroShellOS dares to place users at the center—not just as operators, but as the final authority over their system and their data.

Flexible Operating Modes

NeuroShellOS is not built for one type of user—it is built for all. Whether you're a beginner exploring Linux for the first time, a power user building complex workflows, or a developer fine-tuning AI behaviors, the system adjusts to your comfort, skill, and intent.

At its core, NeuroShellOS supports **multiple operating modes**, each tailored to a different kind of interaction:

- **Manual Mode**: Ideal for users who prefer full control. All actions are initiated by the user, with traditional CLI or GUI workflows. The AI remains passive, only offering suggestions if asked.
- **Semi-Automated Mode**: For users who want intelligent assistance but retain decision authority. The assistant can suggest actions, pre-fill configurations, summarize logs, or prep environments—but nothing is executed without confirmation.
- **Fully AI-Assisted Mode**: Enables the assistant to take broader initiative—performing trusted tasks automatically, resolving common system issues, or adjusting settings based on behavioral patterns. All key actions remain visible, explainable, and reversible.
- **Custom Mode**: Offers complete user-defined control. Users can mix and match features from any mode—deciding which parts of the system are automated, which remain manual, and how the assistant should behave in specific contexts. Preferences can be saved as profiles and switched dynamically.

In addition to interaction modes, **performance-based execution modes** further refine system behavior based on available resources and user priorities. These include:

- **Low Resource** – Minimizes system resource usage; ideal for lightweight systems and battery conservation.
- **Low Resource Vega** – Uses cloud computing wherever possible to offload processing and preserve device performance.
- **Mid Performance** – Balanced mode for stable multitasking with moderate local AI and assistant features.
- **Mid Performance Vega** – Enhances responsiveness through cloud integration while maintaining efficient local performance.

- **High Performance** – Unlocks deeper system intelligence and responsiveness with heavier local AI operations.
- **High Performance Vega** – Combines high local AI processing with scalable cloud support for optimized throughput.
- **Ultra Mode** – Enables full system capabilities for maximum AI assistance, responsiveness, and customization.
- **Ultra Vega Mode** – Maximum performance with aggressive use of cloud services to accelerate AI and background models.

The **"Vega" variants** are cloud-augmented profiles that dynamically shift heavier AI logic and background tasks to the cloud—**reducing local hardware strain while maintaining full functionality**. These are especially useful on devices with limited memory or processing power, allowing advanced features without compromise.

All modes—whether behavioral or performance-based—are **modular and reversible**. Users can switch between them at any time, apply them to specific workflows, or configure transitions based on power, temperature, or network conditions.

The interface and tooling are also **dual-layered**: GUI dashboards and CLI interfaces coexist and interact with shared intelligence. Whether through visual tiles or shell commands, the experience adapts without friction.

Furthermore, the system is **environment-aware**. It detects when the user is offline, resource-limited, or working in secure environments and automatically scales down background services. Once resources return or connectivity is restored, extended capabilities reactivate smoothly.

Some **advanced optimization features and extended mode capabilities** depend on the chosen edition of NeuroShellOS. These are detailed in the **Editions section**, where each system build is tailored for different use cases and performance tiers.

In NeuroShellOS, flexibility is not an extra—it is the foundation. The system transforms itself to fit the way each user works, thinks, and evolves.

LLM and Helper System Architecture

NeuroShellOS is architected around a multi-layered intelligence model designed for high performance, modularity, and complete user control. At the core lies a **purpose-trained Linux-focused LLM**, tightly integrated with a set of **on-demand helper modules** and lightweight ML components—all coordinated through a minimal resource management system.

Core LLM for Linux Intelligence

The system includes a compact yet capable LLM trained exclusively on Linux system-level tasks, commands, and developer workflows. This model is designed to operate fully offline, delivering fast and context-aware assistance across both GUI and CLI environments.

- **Training Scope**: Linux-specific knowledge base
- **Token Size**: Approximately 80M to 120M tokens
- **Format**: Optimized for efficient local inference
- **Quantization Support**: Full-precision or quantized variants, auto-selected based on system performance
- **Runtime Engine**: Powered by the best-performing local inference engine available for the device

This LLM setup reflects the **base configuration** of NeuroShellOS. However, the **integration, features, and deployment options may vary between editions**:

- During installation, users may choose from different LLM builds depending on their system's capabilities—selecting lightweight, balanced, or full-performance variants.
- Alternatively, users may opt to install **all LLM performance levels**, enabling dynamic switching depending on workload or resource conditions.
- LLMs and associated AI packages are retrieved via **edition-specific repositories**. For example, a desktop edition will only have access to components, tools, and LLMs tailored to its performance and interface class.
- **LLMs can be changed, replaced, or updated—but always within the defined scope of the installed edition**, ensuring system integrity and compatibility.
- These repositories may include expanded features, performance-tuned versions, or updated models—strictly filtered by edition identity.

Edition boundaries are maintained to avoid unnecessary complexity or incompatibility. This safeguards system stability while still allowing deep customization inside each edition's sandbox.

AI Engine (Interface Controller)

At the heart of the system's intelligent behavior lies the **AI Engine**, also referred to as the **AI Interface Controller**. This is the central coordination layer that governs all AI-driven components, user interactions, and intelligent decisions within NeuroShellOS. It acts as the unified bridge between the core LLM, helper scripts, modular ML models, system logic, and user-facing interfaces (both GUI and shell).

It is responsible for:

- Interpreting and processing user prompts or system events
- Routing tasks to the appropriate intelligence layer (LLM, ML, script, service)
- Managing all AI-related input/output across the OS
- Handling **all communication with cloud-based AI modules**, when enabled
- Providing a secure, extensible control path for AI-driven operations
- **Including and managing the lightweight Orchestrator** that controls on-demand execution of scripts and modular ML models

The **Orchestrator** is an internal subsystem of the AI Engine. It is responsible for detecting when lightweight helpers or background ML components are required, launching them only when needed, and releasing system resources once their task is complete. It operates entirely under the AI Engine's control.

Edition-Specific Behavior

The structure and capabilities of the AI Engine differ across NeuroShellOS editions:

- In offline-first or lightweight editions, the AI Engine is minimal and optimized for fast, local interaction
- In full-featured or cloud-enhanced editions, it enables advanced routing, persistent background logic, context memory, voice interfaces, cloud sync, and modular extension support
- The AI Engine also determines which version of the orchestrator and assistant modules are loaded, based on edition constraints

Even though the engine remains consistent in core function, its **complexity, depth, and feature access are strictly edition-aware**.

Extensibility & Control

The AI Engine supports full user configurability:

- Basic and Advanced modes offer toggle access to assistant depth, automation, and interaction scope
- Developer Mode enables power users to modify routing logic, inject or replace modules, and build deeply customized experiences
- The controller handles all AI-related triggers, context routing, cloud permissions, and system integration points—ensuring a secure and predictable environment

Through this unified interface, NeuroShellOS achieves **true AI-native system structure**. Whether offline or hybrid, minimal or powerful, every AI operation in the system—LLM execution, helper activation, script automation, or cloud cognition—flows through the AI Engine and its orchestrated intelligence pipeline.

Beyond Linux: Cross-Platform Plans and Specialized Variants

While NeuroShellOS is primarily built on a Linux foundation, its core architecture and AI-driven infrastructure are designed to be extendable far beyond traditional desktops and laptops. The system's core concepts—offline AI integration, script-based automation, modular ML components, and a flexible orchestrator—can be adapted to run on other platforms with tailored optimizations.

One of the most promising directions includes a **dedicated Android-based variant**, currently being conceptualized in **three editions**:

- **Gamer Edition**, focusing on performance, tuning, and AI-assisted controls
- **Hacker Edition**, optimized for terminal access, networking tools, and developer-grade control
- **Normal Edition**, intended for daily use, smart assistance, and AI-supported messaging, browsing, and utilities

These Android editions are planned to offer a distinct experience compared to traditional Android builds, with an AI-native overlay, unique GUI, and tightly integrated assistant support. However, due to performance and compatibility requirements, these editions are **targeted only at select Snapdragon-based devices** capable of handling onboard AI tasks or bridging with the cloud.

Another lightweight deployment is planned for **Raspberry Pi boards**, where the system architecture takes a different path: instead of a full LLM running locally, the design emphasizes **on-demand ML models**, lightweight scripts, and orchestrated behaviors to **simulate LLM-level intelligence** in a highly optimized way. These builds will be ideal for edge computing, local automation, or intelligent IoT tasks with limited resources.

Looking even further ahead, a conceptual hardware platform inspired by **ESP32-class microcontrollers** is being planned. This custom microcontroller—or low-power board—would be **purpose-built for running NeuroShellOS-like behavior**, with embedded ML capabilities, a real-time orchestrator, and optional cloud fallback. Although still conceptual, this idea reflects the goal of making AI-native OS architecture **accessible even in ultra-constrained environments**.

In all cases, the goal is not just platform portability, but **platform-specific intelligence**, where each build uses the available resources—whether it's a high-end SoC, a Raspberry Pi, or a custom chip—**to deliver an adaptive, AI-enhanced, privacy-aware experience**.

User Editions

The editions listed here represent only a few sample categories. Many more specialized or hybrid editions can be built in the future based on user needs and hardware capabilities. Each edition has its own unique set of features such as security configurations, GUI customization options, performance tuning, hardware-specific optimizations, etc.—each tailored to the edition's intended purpose and tier. While only the main, explicitly defined features are mentioned here, every edition includes additional underlying enhancements such as hardened modules, access control policies, GUI-level restrictions, custom integrations, etc., based on its role. These ensure that each edition is secure, responsive, and purpose-optimized. Minimum system requirements, included tools, and feature scope will also vary between editions. If any tiers are listed, they include all performance modes described in the Key Features Overview section. However, each mode will operate within the maximum and minimum limits defined by that specific tier's capabilities.

Desktop Edition – Tiered Structure Overview

The **Desktop Edition** is available in four performance-based variants—**Base, Tier 1, Tier 2, and Tier 3**. All tiers include the **full feature framework**, including the core LLM, the AI Engine (Interface Controller), orchestration logic, and assistant modules. However, each tier differs in the **depth of integration, system-level enhancements, and pre-configured capabilities**. Importantly, users can **upgrade or extend any tier** using official repositories, updates, or edition-specific packages.

Base

The Base version acts as a lightweight **software-mimic layer** of the full system. While it contains all core AI logic—including the full LLM, orchestration control, and GUI access—the rest of the components (like ML models, background security tools, and advanced modules) are kept minimal or disabled by default. Its AI Engine operates with general-purpose logic, with fewer system-specific optimizations. This edition is ideal for legacy hardware, basic users, or those who want to customize from a minimal starting point.

Base includes structural placeholders and compatibility for all features available in higher tiers, allowing users to add them later via updates, package repositories, or performance upgrades.

Tier 1

Tier 1 builds directly upon Base, enhancing it with **an optimized AI Engine** that includes **smarter execution paths, reduced latency, and adaptive performance tuning**. All major features from Base are retained, but with improved responsiveness and slightly deeper

integration. Think of it as the same software but with **built-in efficiencies**, suitable for mid-range devices or users who want a smoother experience with minimal setup.

Tier 2

Tier 2 adds its own **capability layer** beyond just performance. It introduces **security-oriented modules**, including a **lightweight and extendable local threat database**, integrated threat detection scripts, and assistant-supported defensive automation. This makes the edition suitable for users concerned with integrity, basic malware protection, or offline monitoring. The AI Engine here gains additional control hooks tied to system state, shell monitoring, and safe automation. GUI customization, assistant overlays, and system diagnostics are more deeply tied into user context.

Tier 3

Tier 3 delivers the **highest-level Desktop Edition experience**, merging full-system intelligence with AI-driven security, automation, and adaptive control. It includes everything from the lower tiers and adds:

- Expanded threat defense capabilities (advanced rule sets, customizable filters, signature updates)
- Full-scale assistant overlay support with visual system feedback
- Advanced scheduling, automation triggers, and AI-based optimization of tasks
- Deeper local + cloud orchestration switching
- User-controlled AI autonomy levels (prompt-based, proactive, hybrid modes, **Full Autonomy Mode(will be for advanced users)**)

Tier 3 is ideal for high-end desktop users who want total control, rich AI automation, and in-depth system introspection.

Backward Compatibility & Tier Flexibility

The **NeuroShellOS tier system is designed to support both upward and downward transitions**. While any lower-tier system can be upgraded to a higher tier by adding components (through package repos, updates, or edition installers), **downgrading or reverting to a previous tier is also possible**—as long as the user confirms removal of the higher-tier modules, configurations, and dependencies.

The OS maintains **tier-aware system states**, allowing rollback or reconfiguration without needing a full reinstall. This flexibility ensures that users can scale the OS **up or down based on their hardware, preferences, or resource availability**, without breaking core functionality.

Developer Edition (Programmers, Builders & Tinkerers)

The **Developer Edition** of NeuroShellOS is designed for programmers, open-source builders, and system engineers who want deep control over their environment while enjoying intelligent, assistant-enhanced workflows. It includes all core components from the Desktop Edition, along with expanded developer-specific capabilities such as AI-powered shell and GUI integration, live feedback systems, and smart automation tools for coding, debugging, and deployment.

A major highlight is the **plugin-based integration with external development tools and IDEs**. The system's AI Engine can interface directly with supported editors (such as VSCode-style environments), offering features like smart code suggestions, dependency automation, AI-assisted refactoring, and shell command augmentation—turning development environments into intelligent assistants.

The Developer Edition is available in **five capability tiers**, each refining performance, tooling depth, and system-level integration:

- ### Base
  Includes essential CLI tools, assistant overlays, and **mimic-style versions** of advanced features like debugging and automation. Optimized for low-resource systems or minimalist setups.
  This tier **removes non-essential Desktop components** such as media dashboards, document viewers, and other general-use UI layers—ensuring a clean and focused development environment.

- ### Mid
  Enables full versions of development features: plugin support for editors, AI-guided shell assistance, context-aware script generation, and **basic system optimization**. Introduces **light security hooks**, system auditing, and responsive scripting tools.

- ### High
  Adds more advanced **system-level security tools**, including live process monitoring, sandboxed script execution, and rule-based AI logic gates. Also includes **Desktop Edition features** that are specifically useful for developers, like GUI automation helpers, quick-access overlays, and live feedback panels.
  Adds **localhost testing**, sandboxed environments, and modular VM layers for isolated development and testing workflows.

- ### Ultra
  Delivers **advanced AI capabilities**, multi-project context awareness, custom orchestration APIs, and full-scale automation pipelines. Comes with intelligent profiling, real-time optimization assistance, and adaptive assistant workflows.

Consumes more system resources, but significantly improves development speed and automation depth.

- Epic *(Experimental)*
  Pushes toward **self-correcting development**. The system interprets intent, executes tasks automatically, and refines results in real-time—building and correcting code or environments until the desired outcome is achieved. Built for cutting-edge experimentation.

All tiers share the core LLM, plugin architecture, and orchestration controller. Security and optimization features begin at **Mid** and expand in scope through **High**, **Ultra**, and **Epic**. Each tier builds on the previous while maintaining **backward and forward compatibility(except for the Epic tier**, which is isolated due to its experimental nature and not guaranteed to support tier rollback or forward-porting**)**, allowing developers to move across configurations dynamically via updates and repo modules.

Security Edition

The **Security Edition** of **NeuroShellOS** is engineered to support the entire lifecycle of cybersecurity operations—from reconnaissance and penetration testing to digital forensics and live defense analysis. It is not just a collection of tools, but a thoughtfully constructed operating environment that supports intelligent workflows, rapid deployment, secure isolation, and minimal configuration overhead.

Unlike traditional security-focused OS distributions, this edition is **pre-configured with custom drivers**, **controlled system services**, and **security-aware behaviors** from the ground up. From the **Base tier**, it provides a stable and ready environment that includes elements drawn from the Developer Edition when directly useful—such as low-level debuggers or interpreter support—while intentionally excluding heavier components like full-stack development environments or UI builders.

Every tier focuses on clarity, control, and precise adaptability for security professionals. The system is lean, intelligent, and fully optimized for offline or air-gapped use—but can scale into AI-assisted simulations and controlled cloud interactions in higher tiers.

- Base
  A secure, minimal foundation with essential shell utilities, forensic-ready tooling, and hardened driver and service defaults. The environment is locked down for testing and exploration, offering only the developer-grade components necessary for introspection or script logic. It is designed for USB/live use, clean lab setups, or hardened installation.

- ## Mid
  Expands the toolset to include passive network scanners, data extractors, local audit tools, and lightweight scripting assistants. A **small optimization engine** is introduced in this tier: when activated, it monitors for system strain during intensive tasks (e.g., heavy network captures, large file inspections), and can **automatically free resources by disabling unnecessary services or GUI elements temporarily**. This engine is lightweight, uses minimal RAM, and requires no manual tuning—perfect for laptops or constrained devices. Basic AI assistance is also introduced for tool usage guidance and setup tasks.

- ## High
  Adds intelligent resource isolation for local execution environments and support for disposable sandboxed test setups. Users can simulate exploits or inspect payloads in system-controlled containers with defined memory, storage, and driver constraints. Adds in debugging and reverse engineering utilities from the Developer Edition as needed but avoids full development stacks. Supports persistent lab state management and limited automation.

- ## Ultra
  Unlocks orchestration for full-scale red/blue simulations, AI-guided adversarial modeling, and network-wide behavior tracing. From this tier, users gain access to:

  - Fully configurable disposable VMs
  - Driver and kernel module control at the sandbox level
  - Memory and compute capping
  - Scripted or AI-triggered scenario execution
  - Intelligent monitoring of toolchain usage and performance
    VM behavior can be predeclared (headless or interactive, isolated or connected), with optional on-demand provisioning from a curated tool repository.

- ## Epic *(Experimental & Standalone)*
  Designed for high-end research and autonomous testing environments, this tier enables advanced logic workflows, including AI-guided exploitation simulations and scenario orchestration from minimal input (e.g., target IP or attack surface descriptor). All operations are confined within secure, AI-managed sandboxes. Epic includes deep scripting and binary analysis capabilities derived from the Developer Edition but remains interface-light and focused solely on security execution. Due to its autonomy and isolation architecture, it is deployed separately and cannot be dynamically enabled or downgraded.

NeuroShellOS Security Edition is built not merely to provide tools, but to create an intelligent, responsive operating environment for real security work. Every layer, from drivers to orchestration, is configured with cybersecurity tasks in mind.
It shares only the **most relevant developer features**, introduces a **light optimization layer** in Mid tier, and offers **fine-grained control over system behavior** across all levels.

The entire tier system—from **Base to Ultra**—is designed to be **fully forward- and backward-compatible**, allowing users to upgrade or downgrade between tiers based on need or hardware capacity. Only the **Epic tier** stands apart as a self-contained, experimental tier, due to its advanced orchestration logic and isolated architecture.

Education Edition

The **Education Edition** of **NeuroShellOS** is purpose-built for learning—centered around **computer science education** but also supporting a broad spectrum of **general knowledge and digital literacy** needs. This edition offers a distraction-free, intelligent environment for schools, universities, online learners, and technical bootcamps.

From the **Base tier**, users have access to curated knowledge, beginner tools, and a clean OS environment ideal for introductory computing. As users progress through tiers, they gain increasing levels of AI-powered interaction, file-based learning support, and content-driven automation. The OS is designed to **grow with the learner**—offering personalized help, secure experimentation, and seamless performance on even low-spec systems.

- ### Base
  A stable, lightweight setup ideal for basic education. It includes essential tools such as word processors, code editors, and preloaded general knowledge modules covering computing basics, digital safety, and logic concepts. It emphasizes simplicity, security, and accessibility.

- ### Mid
  Introduces enhanced learning features powered by a local AI engine. From this tier:

  - Students can upload **PDFs, TXT files, or structured documents** to extract summaries, generate notes, or ask questions about the content.
  - The system transforms these materials into interactive learning experiences.
  - If **cloud AI support is enabled**, the assistant provides **richer analysis, deeper summarization**, and adaptive questioning based on the uploaded content. Teachers benefit from lesson planning tools and content-based AI suggestions to assist in lectures or assignments.

- ### High
  Offers project-based coding environments with sandboxing, curriculum

customization, and support for safe student experimentation. Includes offline lab modules, structured code tutorials, and document-to-exercise generation. Teachers gain full control over classroom modules and progress analytics, including support for multiple file formats (Markdown, CSV, HTML).

- Ultra
  Enables AI-curated learning tracks, real-time collaboration tools, performance dashboards, and integrated assessment engines. Educators can convert entire curriculum documents into labs, practice tests, or review assignments with minimal manual effort. The AI can even recommend content paths based on learning behavior and historical interactions.

- Epic *(Experimental & Controlled Use Only)*
  Built for advanced research and institutional education. This tier supports:
  - Automated lab setup from academic papers, research files, or structured PDFs
  - AI-assisted grading and feedback loops
  - Intelligent synthesis of reading materials and structured learning flows
    Epic is an isolated tier and designed for supervised deployment only.

From the **Mid tier onward**, NeuroShellOS Education Edition empowers students to learn not only through applications, but directly from their own documents. With just a file upload—be it a textbook PDF or a handwritten scan—students can receive summaries, generate notes, or create exercises.
And with cloud AI enabled, results are even more adaptive, context-aware, and detailed.
All standard tiers—**Base through Ultra**—are fully **forward- and backward-compatible**, allowing seamless upgrades or tier changes. The **Epic tier**, due to its autonomous structure, remains isolated and not upgradeable from other tiers.

Creative Edition

The **Creative Edition** of **NeuroShellOS** empowers individuals and teams working across artistic disciplines—**writers, visual designers, audio producers, editors, animators, and multimedia creators**. Built on a performance-optimized Linux base, it merges reliability with creative flexibility, offering responsive assistance, automation, and intelligent tools for varied creative needs.

This edition prioritizes modular tools, aesthetic interfaces, and low-friction workflows. While it avoids default resource-heavy generation models, it gives users the ability to **scale up creative AI** only when needed—either via installed models or via cloud-connected options.

- ## Base
  A minimalist, focused workspace supporting markdown writing, sketch viewing, audio journaling, and structured project planning. It emphasizes visual clarity, reduced noise, and clean file-based workflows for creators who want speed and simplicity.
- ## Mid
  Introduces **lightweight creative AI support** including:
  - Prompt generation, naming help, and structural outlining
  - Auto-tagging and organizing assets
  - Text-based editing guidance and captioning
    By default, **heavy creative generation (image/audio/video)** is not available at this tier.
    However, if **cloud AI support is enabled**, users may optionally access generation features through remote resources—allowing **Mid-tier systems to create** complex content **without local model burden**.
- ## High
  Adds support for creative pipeline scripting, templating, and enhanced AI co-editing. The assistant can:
  - Analyze multimedia drafts
  - Offer suggestions across mediums (text, voice, layout)
  - Track versions and patterns for reuse
    Still optimized for **resource-conscious environments**, but also cloud-enabled for expanded generation access if permitted.
- ## Ultra
  Unlocks **native multi-modal generation features** such as:
  - Image generation
  - Music or audio synthesis
  - Animated storyboard generation
    These run either locally (with higher-spec hardware) or via configured cloud resources. The assistant can now fully participate in the creative cycle—generating, refining, and integrating outputs in real-time across supported tools.

- Epic *(Experimental & Controlled Use Only)*
  A sandboxed, high-performance environment for AI-driven creativity labs and production studios. Offers:
    - Multi-agent synthesis and remixing
    - Generative orchestration of projects
    - Distributed inference for large-scale creative workflows
      Epic remains isolated from lower tiers due to its experimental infrastructure.

NeuroShellOS Creative Edition is designed to be flexible. By **default**, heavy generation tasks are reserved for **Ultra and Epic tiers** to preserve performance.
But if **cloud AI is enabled**, even **Mid or High** tiers can access generation capabilities—without the need for local model hosting.
All standard tiers (Base through Ultra) are **forward- and backward-compatible**, while **Epic** operates independently for high-end, experimental use cases.

Gamer Edition

The Gamer Edition of NeuroShellOS transforms Linux into a high-performance, AI-assisted gaming environment tailored for both casual and competitive needs.

While gaming on Linux is still evolving, it already shows immense potential—especially when it comes to resource efficiency. Unlike traditional OS platforms that often carry heavy system loads in the background, a well-optimized Linux environment can achieve higher performance with significantly lower resource usage. This becomes especially impactful when Linux is used solely for gaming: eliminating unnecessary services, minimizing overhead, and offering far greater control than typical Windows setups.

NeuroShellOS Gamer Edition builds on this advantage, using a carefully constructed architecture that maximizes performance, reliability, and user freedom. It intelligently handles WINE and Proton layers to ensure seamless execution of non-native titles while offering manual or AI-assisted optimizations.

Critically, this edition is built only on tested, stable base systems and kernels selected specifically for low-latency gaming, Vulkan readiness, input responsiveness, and I/O speed.

Whether you're tuning your system manually or relying on the assistant to configure your drivers, shaders, and launch parameters, the Gamer Edition provides true flexibility, precision, and an OS that feels purpose-built for play.

It features low-latency controls, game-specific optimization profiles, and optional AI automation—all governed by the user and adaptable to every level of gamer, from casual players to competitive streamers.

Gaming Mode

A central feature of all tiers is the **Gaming Mode** toggle:

When enabled, it suspends non-essential components like the **AI Core**, **Orchestrator**, and background analytics to dedicate system resources fully to gameplay.

Common Features Across All Tiers

- Game Center
  Unified launcher and dashboard for native, emulated, and cloud games
  - **AI-based game detection** for automatic setup
  - **Manual configuration options** for advanced users
  - Optimization profile management per title
- Low-Latency Network Manager
  Real-time tuning of network behavior for gaming (ping reduction, packet shaping)
- Overlay System
  FPS, input latency, thermals, and resource usage visualized live during gameplay
- Input & Controller Suite
  Support for remapping, latency reduction, and AI-assisted calibration
- Performance Scheduler
  Dynamically balances system load to prioritize game threads and GPU usage
- Advanced GUI Panel for Gamers
  Lets users toggle Gaming Mode, control overlays, tweak system optimization levels, and manage AI involvement
- AI Assistant **(when enabled outside of Gaming Mode)**
  Assists with shader caching, game modding, driver updates, and launcher configuration,installation

Tiers

Zerco ***(Core Gaming Shell)***

- No runtime AI during gameplay
- Mimic-based static optimization
- Advanced manual network and resource configuration
- Best for stable, low-resource or competitive use

Hashi ***(Performance Optimization Tier)***

- Active optimization engine
- AI assists pre-game setup, then engine handles live performance tuning
- Adaptive system resource routing and real-time thermal control

Alphadeco ***(Creator + Automation Tier)***

- AI-managed modding, game streaming setup, and content assistance
- Optimizer + hybrid runtime mod manager during gameplay
- Designed for content creators and streamers

Tier Compatibility and Transitions

- All Gamer Edition tiers are built with **forward and backward compatibility** in mind.
- A **Zerco-tier system** can be upgraded to **Hashi** or **Alphadeco** without reinstallation—using modular update layers or via apt-repo and assistant-led setup.
- Similarly, higher tiers can revert to minimal modes by disabling features or switching profiles manually or via AI.
- Optimizers, configs, and core environments are **aware of lower and higher tier schemas**, ensuring smooth transitions across upgrades.

Whether you prefer a clean, no-overhead environment or a powerful AI-assisted gaming setup, **NeuroShellOS Gamer Edition** adapts to your preferences and performance requirements.
With **Game Center**, manual control, tier flexibility, and intelligent configuration systems, it builds a Linux experience that understands what gamers actually need.

Privacy Edition

The **Privacy Edition** of NeuroShellOS is designed for users who prioritize digital sovereignty, layered security, and full control over AI activity. Through strict modular separation and privileged orchestration, it ensures that no component—AI or human-facing—can operate outside its permitted boundaries.

From user prompts to AI-assisted automation, every action is traced, policy-gated, and hardened with forensic-level protections. The system is intended not only to withstand compromise but to actively neutralize, log, and recover from intrusion attempts.

This edition draws structural and architectural inspiration from security-focused systems like **Qubes OS**, particularly in its modular compartmentalization of trust zones. However, NeuroShellOS extends the concept further with integrated AI governance, encrypted hardened layers, and responsive privacy mechanisms.

System Architecture Overview

1. *AI Core*

- Hosts the local LLM and supporting lightweight ML models.
- Operates entirely on-demand under AI Interface Controller approval.
- Cannot access network or system-level resources independently.
- Supports analysis, scanning, and summarization for security and diagnostics.

2. *Network & Security Layer*

- Provides a fully segmented networking stack with packet filtering, DNS hardening, and outbound control.
- Cannot be accessed directly from the GUI or AI Core.
- Logs all connection attempts and validates protocol behavior.
- Communicates with the Hardened Core through trusted pathways only.

3. *Security + Hardened Core*

The **Hardened Core** is the highest-security zone within NeuroShellOS, responsible for storage, monitoring, and protection against both internal and external threats. It is architected to remain isolated from other domains and enforces zero-trust interaction principles throughout.

This module is purpose-built for tasks such as:

- Securely storing encrypted user data
- Performing low-level integrity checks and malware analysis
- Executing internal audits and collecting logs from the system
- Managing and validating its own security updates
- Supporting threat recovery procedures and audit trails

Standalone Boot Mode (Isolated Execution Mode)

The Hardened Core can be booted separately from the main OS using a predefined boot key (e.g., F9). This mode is specifically designed for recovery, manual inspection, or defensive tasks, even if the main OS or GUI is compromised.

In this mode:

- Only a minimal, hardened GUI is presented for interactions
- No command-line access, scripting, or runtime execution is allowed

- Network access is handled via a dedicated isolated driver, enabled only as needed
- USB drivers are supported exclusively to allow users to copy encrypted files out of the hardened environment
- No external writes are permitted back into the system through USB or network
- Updates related to the hardened subsystem or integrity tools can be initiated and applied internally

This design allows users to retrieve important secured data or trigger recovery actions without exposing the system to new threats during a compromised state. All outputs, including decrypted results, system reports, or file copies, remain under strict output policy control.

The Hardened Core maintains its independence from the AI Engine and the GUI, ensuring that critical operations are never exposed to unintended automation or misused access paths.

4. *GUI + Terminal Layer*

- Main user interaction zone.
- Provides antivirus tools, assistant integration, secure file upload interface, and OS customization tools.
- Cannot execute system scripts or touch privileged files without policy.
- Capable of provoking updates, scans, or recovery actions from the Hardened Core via AI Interface Controller routing.
- Users may securely transfer files from the GUI into the Hardened Core's encrypted storage under policy-controlled mechanisms. All transfers are authorized via the AI Interface Controller and require user confirmation to ensure intentional, secure handling.

5. *AI Interface Controller*

- Manages all AI behavior and communications with core domains.
- Ensures that no AI can trigger updates, scans, or file access without policy and explicit routing.
- Can delay or block AI actions during risky or unauthorized operations.
- Logs every decision and tracks all automation states.

Security Update and Maintenance Behavior

- Updates to the hardened modules or AI infrastructure can be initiated via GUI, but are executed internally by the secure subsystems.
- Due to the depth and structure of the privacy and isolation mechanisms, updates may take longer than in standard Linux distros.
    - Each component must be validated, checksummed, and tested before application.
    - Hardened updates are signed, atomic, and verified in isolation before any reboot.
- GUI users will be notified and optionally prompted before such updates.
- Bootable hardened mode also includes a secure offline updater interface, allowing users to plug in signed update images manually.

Summary of Privacy Enhancements

- Hardened Core can operate autonomously and be booted independently.
- No scripts or userland access in hardened boot mode; only secured GUI and update actions.
- Encrypted data can only be inserted from GUI and removed under strict policy.
- Two-tier antivirus across GUI and Hardened Core.
- Updates and encryption are cryptographically bound to the user's installation-time configuration.
- Complete AI governance via Interface Controller with strict resource access gating.

Tiers in Privacy Edition

- *Kryptomania* **Tier**: Base layer offering foundational privacy functions, script-free GUI, and minimal AI control with access to encrypted personal storage.
- *KuKrypto* **Tier**: Mid-level tier featuring integrated AI auditing, enhanced network segmentation, USB key recovery access, on-demand AI security scanning, and moderate system self-healing tools.
- *SeKrypto* **Tier**: Highest level offering fully compartmentalized operation, hardened AI behavior with threat anticipation logic, dynamic forensic logs, isolated boot modes, offline update processing, and full control of encryption key management.

Author's Note

This project is an independent effort and is not affiliated with or endorsed by Qubes OS or its developers.

License:

Author:

Muhammed Shafin P
GitHub: @hejhdiss

Acknowledgments

- The open-source operating system community for foundational principles
- AI/ML researchers advancing accessible and privacy-respecting AI
- Privacy advocates promoting user sovereignty in digital systems
- The broader Linux community for demonstrating collaborative development success
- The **Privacy Edition** of this project is conceptually inspired by Qubes OS.