RICE

Available freely to use, the RICE Operating System (Regionless Inter-Configuration Environment) is a modular, multi-user operating system. It was developed by Kage Yaichiro and various contributors in YE 35. Unlike the majority of offerings on the market, RICE-OS is not only free to use but can be compiled and run on almost any platform.

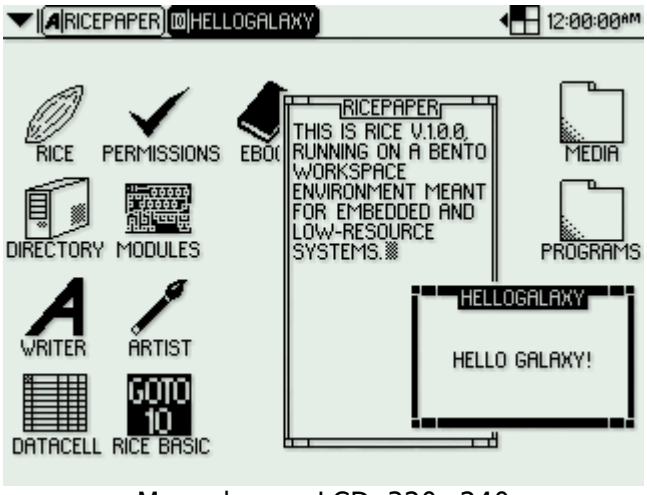
RICE has strong Virtual reality and Augmented Reality interfacing capabilities and I/O capabilities – unique in that it can map and write drivers for hardware plugged into it without requiring one to be written. RICE also plays well with its competitors and can read file-systems and formats from other popular platforms. It uses semantics heavily to optimize itself and more efficiently function, while improving the user experience.

Current Release

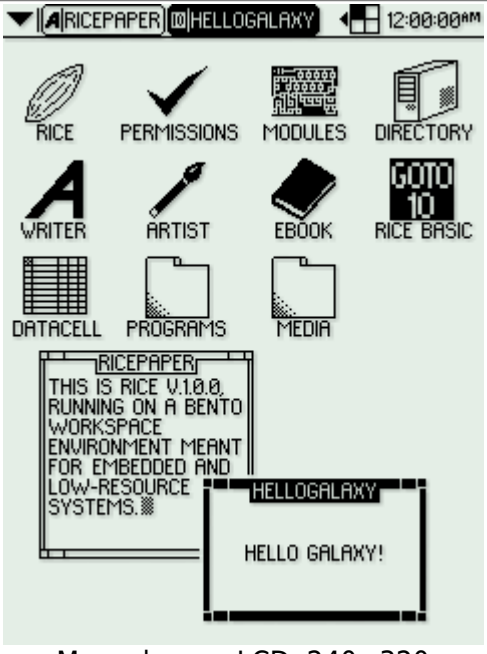
Rice Kernel v1.0.0

Images

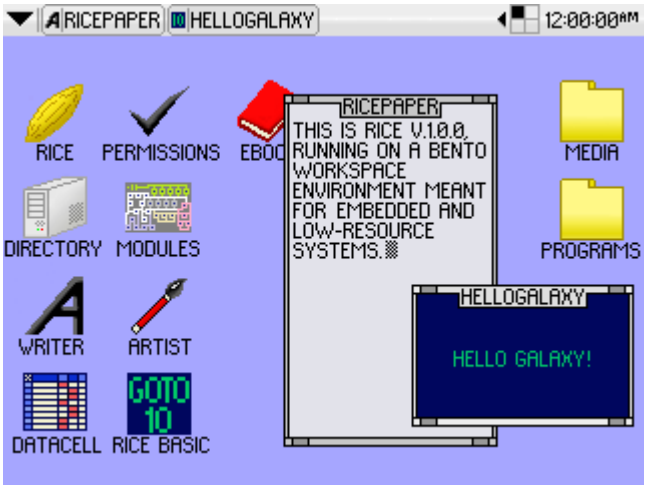
RICE v1.0.0 on Bento Workspace Environment



Monochrome LCD, 320×240

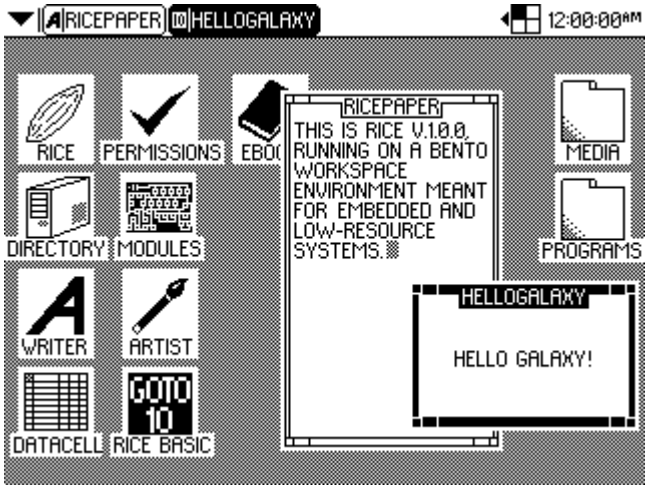


Monochrome LCD, 240×320



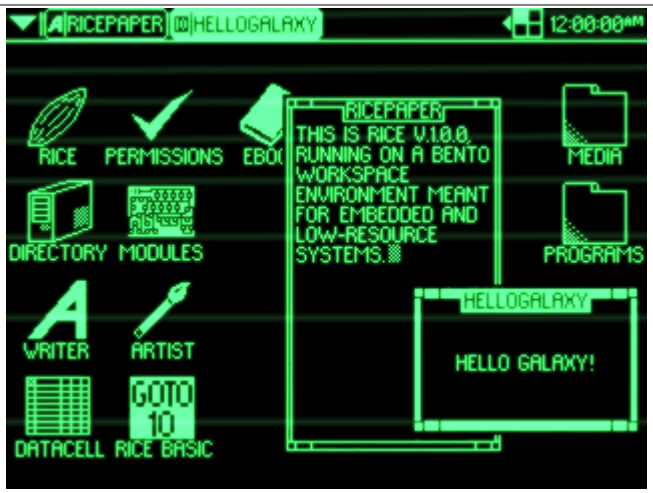
Color LCD, 320×240

Color LCD, 240×320



Emulation, Monochrome with background, 320×240

Emulation, Monochrome with background, 240×320

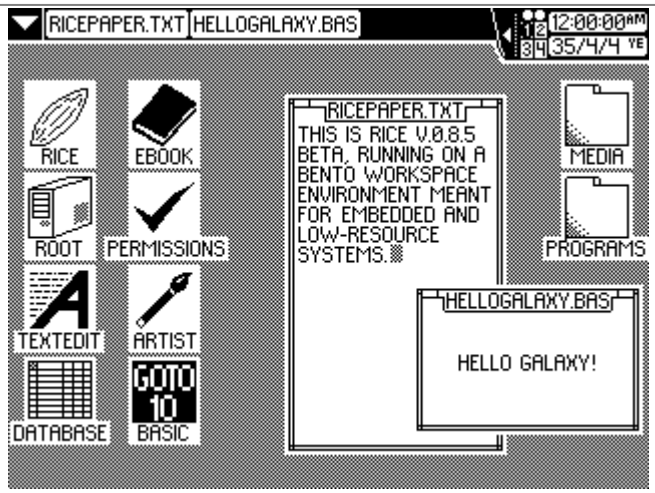


Monochrome Terminal Monitor, 320×240

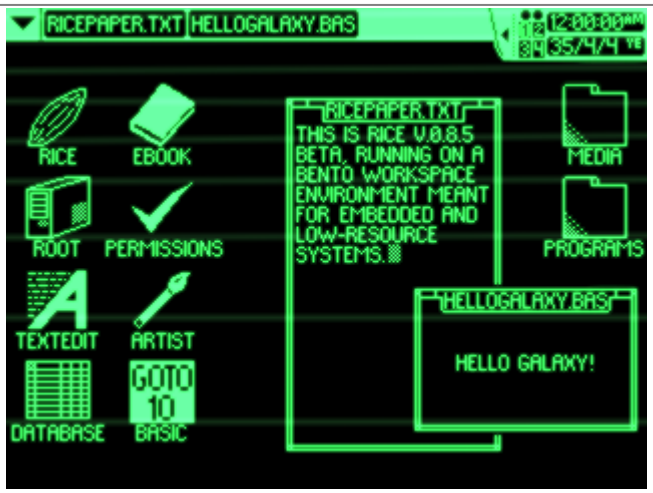
RICE v1.0.2 on Gohan Workspace Environment

1280×720 (click twice for full size)

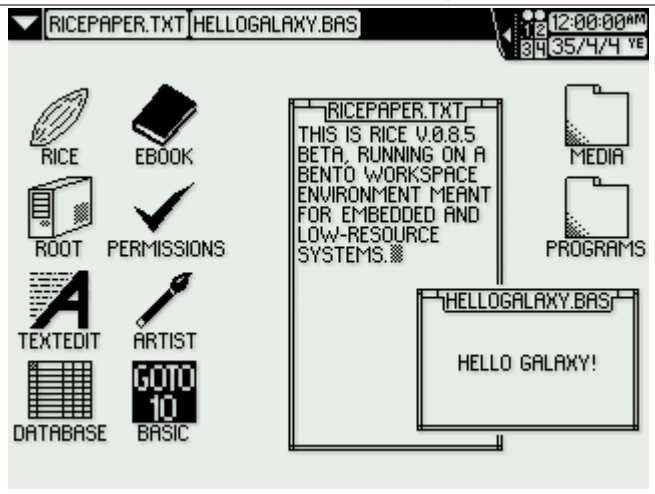
RICE v0.8.5 "Uncooked Rice" on Bento Workspace Environment



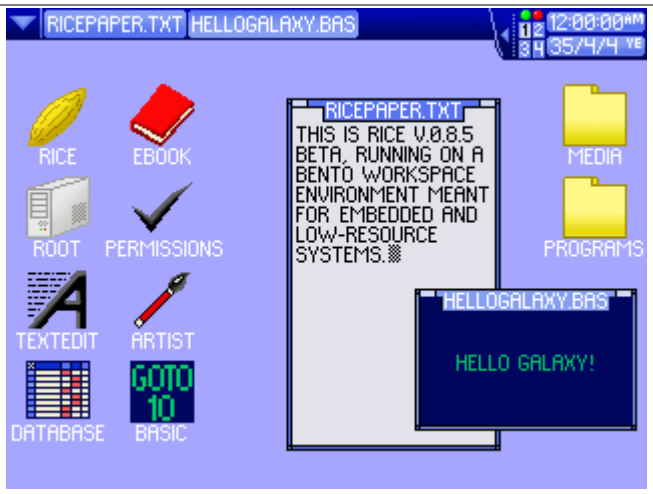
Emulation, Monochrome with background, 320×240



Monochrome Terminal Monitor, 320×240

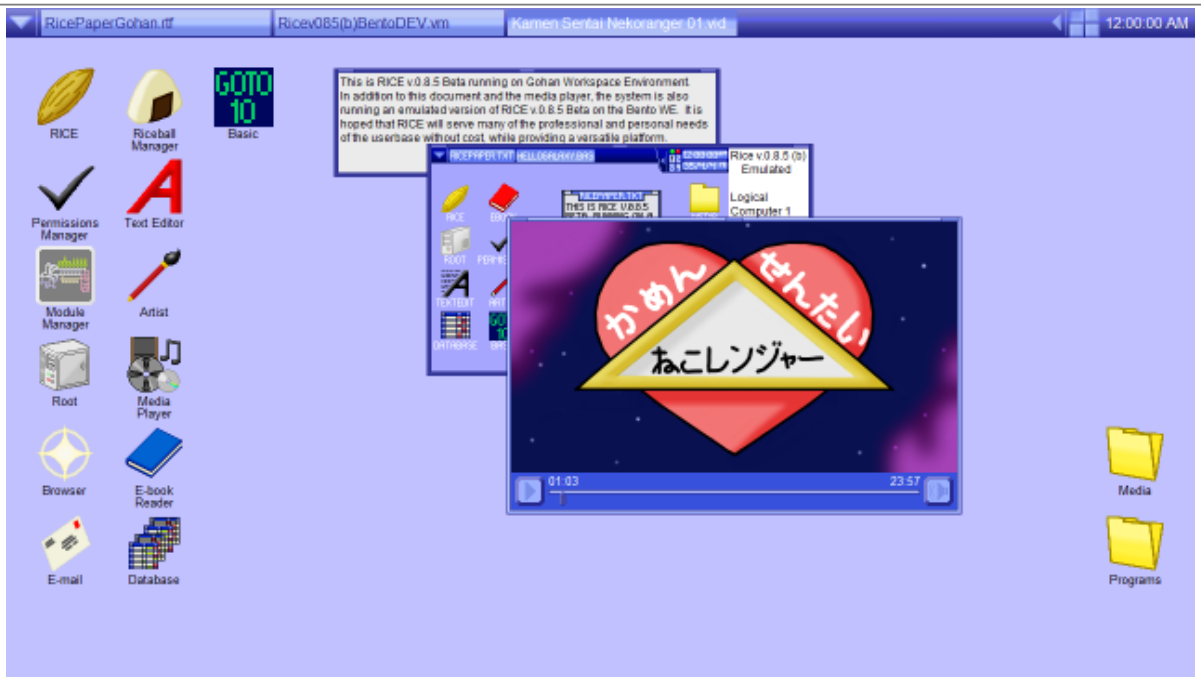


Monochrome LCD, 320×240



Color LCD, 320×240

RICE v0.8.5 "Uncooked Rice" on Gohan Workspace Environment



1280×720

(click twice for full size)

About RICE

Elements of the RICE Kernel and I/O system were developed as early as YE 32 as part of Project THOUGHT, but a different system was eventually chosen which focused more on compatibility with the Mindy platform and was more specialized for military hardware. After additional work through his downtime, Yaichiro had versions of RICE running on assorted hardware including his civilian communicator and his Te-G2 Game Buddy. By then it had evolved, however, into a compact and modular operating system intended for many users and purposes regardless of nationality.

RICE is highly scalable, able to address a limitless number of modules and programs with the exception of the hardware's limitations, and manage their inter-communication and security. The 'small kernel amongst many grains' mentality of the microkernel and module architecture inspired the RICE name, which is in reality a retronym. RICE also is meant to be an important part of a user's tools much like rice is a fundamental Yamataian part of almost any meal, and that it can come in many distributions or “flavors” of RICE. Though the name is somewhat borne of Yamataian culture, the word in trade was chosen to ensure it was not bound to Yamatai. Due to this, RICE tends to function in various capacities from a communicator or game system version to versions capable of managing a spacecraft's functions. Various distributions or “flavors” can be made out of RICE, depending on the needs and modules required for their intended purpose, and functionality can be altered just by placing the needed modules in place rather than by changing the entire system. This also allows a user to customize their RICE installation to utilize what modules are best suited to them.

Design

Boot Process

Hardware Identification, memory hooking

The boot up stages of RICE share some elements with others such as key presses for boot options, though there are differences to make it more capable of identifying threats and details of its environment. It first attempts to use an array of loaders to see what sticks on the hardware its running on, so it can get a foothold and hook into memory.

Self-verification, driver-authoring

Once accomplished, the system runs checksums on its execution code – basically ensuring it hasn't been modified, so no nasty surprises execute on boot (viruses, worms, buffer-overruns and other malware). Verified, the system then identifies what its loaded on and plugged into - verifying what its connected to isn't strange or unexpected. Anything that is new, the system begins using low-voltage resonance signalling to infer the logical pathways of whatever its connected to - and begins the process of writing its own drivers should no existing drivers be found.

Selective element loading

With these early stages complete, the system now moves onto higher level operational topology - basically, deciding what purpose its likely going to be used for and altering its constituents. Now, the system decides that based on its operating environment, to optimize itself. Its base kernel (the very core of the operating system) is fragmented by design into sub-components, which can be loaded and terminated without bringing the system to a whole. Some of these sub-kernels – or Kernel-Modules are specialized for specific hardware or processing tasks. By cherry-picking which it loads intelligently, the OS is changing itself based on what resources it has available – ensuring it performs and scales properly. New modules can be added later but at this point, the system is more concerned with what it **needs** to function on a basic level.

With the basic OS loaded, the system now begins to assess what other purposes it might be used for: Am I running on a neural module? A starship? A car? A starbase? A phone? A watch? Based on this information, the system begins loading its higher decision-making systems - beginning with **required** parts of the Semantics Library – for example, if running on a starship, elements needed to control a ship – or if running in a router, only the elements needed to route and secure information.

From there, the OS begins inferring things on its own more readily from the context of what its used for – that is, user-interaction and the other OS it encounters. It can determine if new libraries need to be added, if security protections need to be removed (for example, if loaded in emulation, not as a native OS). This allows the OS to form compatibility layers on the fly; allowing it to run code meant for other operating systems, run an installation of another operating system like Kessaku OS sandboxed inside RICE, or even allowing this specific RICE installation to run within another operating system. One could even run an installation of RICE within another installation of RICE if they had reason to do so.

Desktop loads

At this point, the system begins loading elements outside of itself. Using input output of the hardware, its linked to to load the Workspace Environment and higher functions. Completed, the user is presented with the interface, ready for use.

RICE Kernel

The RICE Kernel is a microkernel system which relies on modules to properly operate or even communicate with most hardware, but its nature allows it to be robust and high speed while remaining scalable. The RICE Kernel will run, depending on the modules loaded, in scalar, vector, or even quantum modes to best suit the hardware present. This, along with the other assorted modules utilized for the kernel, allow the kernel to manage its resources and conduct inter-process communication that much more efficiently. While IPC is often conducted with the control of the RICE Kernel, trusted modules can communicate together to more directly access hardware and software to minimize resources and cycles.

It differs from most systems in that its I/O systems are handled largely by modules as well, to help facilitate the sheer wealth of options possible. That being said, such I/O functions still have the RICE Kernel's oversight.

Permissions

Depending on the intended role of a part of the system; Module and/or Grouping Permissions may be assigned. These permissions vary in purpose and tend to be numeric in type. Module Permissions are on a sliding scale which by default is set from 0 to 255 in steps of 15. This defaults to 18 standard levels of permissions. While Module Permissions are dictated by magnitude, Group Permissions are special permissions which are applied to specifically manage or lock out users in a more structured hierarchy.

It is difficult to understand, but can be broken down readily. Module Permissions dictate the priority and level of trust various parts of the computer's hardware, I/O, and operating system components have. Group Permissions are a more complex form of permissions which help determine the level of access and trust various other aspects of the system have with each other, and are usually applied to users and networked entities to grant them power over the system in varying degrees. Complexity of Group Permissions can vary widely depending on the size of the network.

Module Permissions

Module Permissions determine how much a module can be trusted to directly connect to other modules without being observed or managed by the kernel. Lower module permissions provide better security and fault protection through monitoring by the RICE Kernel and tend to be lower priority, while higher module permissions allow modules to communicate directly with each other without oversight and preserving resources and speed while also being given a higher priority in the system. These are often carefully balanced and can be adjusted by the program's history and the number of errors in the error log, but can also be adjusted by a user or program with sufficient Group Permissions to do so. This should only be done if the program is not only trustworthy, but is also known to be free of bugs that may otherwise cause crashes.

Module Permissions can be altered by the kernel using semantics and operational history, allowing the system to compensate for security vulnerabilities or buggy code.

Group Permissions

Group Permissions are a robust and specific system of organization which allows a more complex and thorough security policy, particularly for Users, Network Entities such as DHCP and DNS servers or locally networked devices. By placing these entities in groups and restricting various Modules, Programs, Directories, and Files to certain groupings noting levels of access, a more robust security profile is achieved. Sufficient permissions can be used to edit Module permissions for hardware access and optimization, or even Modules themselves. Users in the “Engineering” group can never access or see things intended for the “Payroll” group, be they the programs or network printers for each, unless they are a member of both groups. This vastly simplifies the end user experience as well.

Group Permissions can be altered to a degree by the kernel as well using semantics, but this ability tends to be lesser than with Module Permissions. The computer can temporarily lock out a user, network entity, program, directory, or file and report to the Network Administrator or someone else with high Group Permissions exactly what has happened to cause the issue. Then the operating system relies on them to

alter settings to provide a more lasting solution. Such issues can be tracked and trends determined, allowing the system to show the Network Administrator what issues are repetitive and need resolved.

It should be noted that both Users and individual computers on a network (Network Entities) have their own Group Permissions, so it can be configured that even if a User with credentials logs in from an untrusted computer, their access is not granted because they are connecting from a computer without its own Group Permissions. This is an optional element though, and is not always put in place.

Default groups are Guest (read-only), Everyone, User, Power User, Local Administrator, Network Administrator, and other such elements, though other groups with degrees of permissions can also be made for a network.

Modules

Modules make up the bulk of the operating system, and serve all roles imaginable. Even user profiles and networked computers and printers, for the purposes of the operating system, are considered to be modules. All modules are searchable and any can be represented to the workspace environment if desired along with programs, directories, and files. Every single module has Module and Group Permissions assigned to it, the latter allowing for the editing and changing out of modules and typically requiring Administrator access.

Boot Loader/Guest Virtualization

The Boot Loader Module, as previously discussed, is an important module which gives instructions as to not only how to locate the operating system on the computer, but also gives the user the option to press a key to boot into other modes. It allows for network booting, booting from external drives, flavor-specific instructions, pre-boot authentication, and other elements. This module, however, can be altered to function as an engine for running a virtualized RICE installation on a computer with an existing operating system. Using this, one can run RICE on a computer with Kessaku OS or any other operating system for which a Guest Virtualization Boot Loader is written. The RICE installation will even acknowledge the Host Operating System as a locally networked computer, and can support a host of features.

This usually has a fairly high Module Permission not only because of boot time, but also to give any Host Operating System in virtualization greater control over the session.

Permissions Manager

Permissions Manager is the module responsible for determining the Module and Group Permissions of the system, and has one of the highest Module Permissions of the system. Being a part of the Operating System which is the most sensitive, it also has high Group Permissions to prevent editing by undesired personnel or network entities. Permissions Manager is also one of the most secure and robust parts of the system, due to its being critical enough to be a target of viruses and other threats.

Other variants of Permissions Manager or even supplemental modules are also available, however, to

allow more inter-operation with networked Kessaku OS systems or other elements. These are intended to ease compatibility with other units without compromising security. These all, however, require the consent of someone with high Group Permissions to install.

Hardware Devices

The various pieces of basic hardware such as processors, memory, and internal components are represented by these modules, which are often of the highest Module Permissions. Due to their nature, these modules can be independently assigned tasks as needed by workload, allowing for more robust management of resources and even division into separate logical systems. Parts which fail can also be isolated and kept from causing harm to the system as it runs, or even be powered down for superior power management.

I/O Devices

Input/Output devices are represented in this group, and consist of speakers, monitors, volumetric displays, virtual reality systems, wireless communication, wired communication, infrared and laser communication devices, and various other technologies. The I/O system of RICE is rather robust, capable of interfacing with any number of devices and custom hardware so long as a module is coded for it. Everything from a keyboard and a monitor to a ship's sensors and controls can be connected to this system provided the module exists, allowing RICE to be built in flavors intended for ship control or industrial applications.

This is a throwback to the intended use of the system as a thought-guided system which gave the user full immersion virtual reality feedback for military applications. It is possible for all five senses and even brainwave or wireless technologies to allow telepathic verification and communication, and for Star Army of Yamatai versions to have full support for SPINE. This ability to translate sensory data and control data so directly also lends RICE to use as an integrated and standardized on-board computer platform for cybernetics. Due to the large variety of equipment that can be connected to RICE in this way, and the fact that internet connection devices also fall under Input/Output in RICE, the Module Permission levels are variable with graphics and audio having the highest permissions so as to minimize lag. It is also common for some devices such as Printers to have Access or Group Permissions applied as well to restrict their access.

Workspace Environment

The Workspace Environment is the system by which the system communicates with the end user. A standard installation of RICE will have a text interface (Command Line Workspace Environment), a two dimensional screen with icons and panels (Graphical User Interface/GUI Workspace Environment), or a three-dimensional realm with all manner of icons and panels (Immersive Graphical User Interface/IGUI Workspace Environment). This third option is often used with volumetrics or virtual reality, and tends to be similar in theme and function to the GUI/two dimensional version. They commonly show a degree of integration as well, to the point where they and the Command Line make up a single suite of interfaces.

The default Workspace Environment for RICE has all three of these modes and is collectively named “Gohan”, the Yamataian word for rice or meal. Other environments, however, can be added to achieve various ends. An extremely simple and low resolution/low resource version known as “Grain Pane” is also available. These can alter simply the appearance of the environment, or even change fundamentals of its function. It can be optimized for whatever system or user interface it utilizing it, though the underlying capabilities and structure of RICE will remain.

Common elements include any number of Virtual Desktops which can have specific programs and elements loaded onto each, and the ability to use semantics and available hardware (cameras, touch, volumetric analysis of hands and face) to track a person's optical focus and open up menus only as the person wishes to access them to cut down on delay and menu clutter. This is known as focus tracking, and allows most menus to be removed from a window, replaced by colored window border sections one looks at and uses to perform various tasks be it in two dimensions or three. These can include opening menus, manipulating the window itself, moving it to another virtual desktop, viewing the window's application and name, etc. For legacy's sake, a mouse can also access these menus by clocking on these sections of the border when using the standard GUI. The window border also grows on top when Focus Tracking is disabled, so that the title can be seen and the options can be more readily clicked while still minimizing menu clutter.

Other options include subvocalization, simulated keyboards, and even thought-based control using a mind-machine interface. It should also be noted that the computer can analyze the content of a window, the room and orientation of other windows on a screen or in 3D space, and intelligently size and arrange the windows for the user if desired.

Gohan tends to use 64 by 64 pixel icons, with Arial 10 text on the desktop and Arial 12 text on the taskbar. Icons are usually spaced 32 pixels from any sides and each other to allow room for text. Bento, meanwhile, uses 32 by 32 pixel icons with a minimalist font of capital letters that is 8 pixels high.

This usually has a high Module Permission, to communicate with the I/O system, but it can be lowered on systems which aren't as graphics intensive to allow other modules and programs to use resources.

Active Simulation

Active Simulation is a special module which is dedicated not only to conventional physics computation, but also to helping to sample and emulate sensory, motor, and reflex systems. It can learn from its user to a limited degree to make lifelike environments and has capabilities much like the Civilian Comfort Bed. It often learns how its user reacts to problems to improve its mimicking of reflexes, and and is also capable of using semantics to make lifelike virtual entities. This system is not only useful for the purposes of virtual reality, but also for the purposes of running industrial machinery, controlling ships, and even running embedded in cybernetic limbs.

The Module Permissions, because of the intensive nature of the computations, tend to be set high.

Semantics Library

Semantics is the study of meaning, and represents information which can be communicated by language

or symbolism. The Semantics Library allows RICE the notable ability to not only understand concepts, but also to infer things and comprehend more deeply than the simple reading and execution of syntax and language. RICE is granted advanced comprehension and adaptation abilities far superior to most operating systems save for advanced offerings by this system.

Basic abilities such as converting between written and spoken languages, units of measure, inferring three-dimensional information, semantic searches, and context of format are standard functions on almost all versions of RICE even on scalar processors. This also allows for automated hardware detection and driver writing, adoption of new formats, being able to infer undesirable code and thus protect itself from buggy or malicious software, and the ability to share data and naturally standardize on the most optimal version of code available for a format or driver. It can also more intelligently allocate its resources with semantics. More advanced computer systems can use functionality which is more elaborate, including the ability to understand both verbal and non-verbal cues. It takes a very high-tier vector computer or a quantum computer to comprehend things like body language or sarcasm. There are also protections against the system achieving true sentience at this level, however.

This system has the absolute highest of module permissions.

Language Support

Due to the nature of the operating system being free for many factions to utilize, RICE comes with the ability to load multiple modules for language support. These modules can contain not only text, but also spell check, audio, and comprehension data. Because of this, RICE can easily be used for text to speech, speech to text/dictation, or even translation of text or speech to text or speech between loaded languages. Support for currency and time conversion is also possible. The only limitations are the detail of the language pack.

The Module Permissions vary, but the best language packs tend to have high permissions.

Compatibility Libraries

Libraries from other operating systems can be imported into RICE to allow programs and applications to be run within RICE even if meant for other platforms. The system can use semantics to make patches and improvements to this code as needed to better function with RICE or even evolve as the competing platform does, though there are elements in place to restrict this should it be legally inadvisable. The same semantics which improve RICE's capabilities so greatly can also be used within these applications, permitting added capabilities. Very careful context is taken so as not to compromise security features or copyrighted material in these programs, though much of the improvements RICE makes fall under non-profit derivative works.

These modules tend to have medium to low Module Permissions compared to more essential parts of RICE, but there is some variance.

Virtualization

If a Compatibility Library is not sufficient to the user's needs for some reason, it is possible for modules to be arranged for form logical computers on which a virtual installation can be sandboxed and run. The system can freely execute other installations of the operating system provided they are properly licensed if needed, or even versions of RICE within RICE. In some cases, a very small installation of RICE is used as a boot loader to allow multiple operating systems to share a computer, or even to allow operating systems not intended for a specific type of computer to function on it. This can also be used for testing and security purposes.

These modules tend to have lower Module Permissions than other elements including libraries, but there is some variance.

Thinclient/RDP

Various network capable modules exist, taking advantage of the fact that users and networked devices are seen as entities. Among them are Thinclient and Remote Desktop options, which allow one to connect to another Network Entity/Computer and operate on it. These include security and authentication options, and support many of the features RICE also allows in virtualization. Some custom-made modules will even integrate virtualization, thinclient, and Remote Desktop into a single module.

These modules tend to have lower Module Permissions due to their nature requiring them to communicate with a remote machine, but there is some variance.

Profiles

Profiles are a classification for entities outside the system itself, such as users and networked devices. Group Permissions are applied to them for the purposes of organization and security as previously discussed under Permissions.

Users

Users are rather straight forward. Individual users have information attached to them such as usernames, relevant personal details, passwords, personal settings, biometric data if applicable, whether they are locally or remotely connected, the network entity they are accessing from, and the groups they are members of. A number of elements are stored to aid in semantics as well, though these elements are stored securely to protect them.

Network Entities

Network Entities are external devices connected to the local machine over a network. This can consist of Printers, Workstations, Servers, DHCP and DNS servers, and other devices. Network Entities also have

Group Permissions assigned to them, which can be computed along with User profiles to determine their collective Group Permissions. While the User profile usually determines permissions, it is possible to add security to require the correct combination of User and Network Entity permissions to ensure that the user is connecting from the correct computer terminal.

Supplemental Programs

Supplemental Programs are programs which are not specifically part of the operating system. They tend to be used for all manner of purposes, and are subordinate to the modules and kernel of RICE OS. They are very diverse and include everything from multimedia and text programs to professional-level and programming applications. Different distributions can come with different supplemental programs as readily as they can come with additional modules, and the semantics of RICE allows them to adapt to new formats put forth by the computer industry over time and replicate those changes to other computers.

RICE OS comes with a text editor (text to speech, speech to text, and translation), spreadsheet editor, webpage editor, image editor and viewer, basic audio/video recording and playback software, a browser, a communications and e-mail system, a programming console (which can help the user with context and design using semantics), and assorted other utilities.

Group Permissions assigned to Supplemental Programs are used to limit who has access to them, ranging all the way from rendering them invisible/inaccessible to the user through red/write/copy/delete to being able to edit the program and its code.

Directories and Files

Directories and files share similarities with other operating systems, giving structure to the system. However, they are also capable of Group Permissions and are thereby searchable through it while also having security added to them. This, among other details, allows them to be searched with semantics and with object-oriented searches as well as conventional searches by name and by hierarchy.

A semantic-aided compression format for files and directories is also supported natively by RICE, dubbed the Riceball format. It tends to have an extension of “.rb”. While other computer systems are not natively able to unpack riceballs without semantics and RICE libraries, it is possible for the Riceball to come with a custom unpacker to allow opening on other operating systems by detailing step-by-step how to unpack that specific file in explicit syntax. The typically used format for these compressed files is “.rbu” for Riceball Unpacker. Standard compression formats are also natively supported.

Care should be taken not to assign a file or directory a Group Permission which the Supplemental Program needing to access them cannot access. Group Permissions assigned to Directories and Files are used to limit who has access to them, ranging all the way from rendering them invisible/inaccessible to the user to being able to red/write/copy/delete.

Licensing

The licensing of RICE is Open Source, meaning that it can be freely modified and distributed in various flavors, though the source material should be noted and acknowledged rather than presented as a completely original creation. Programs can also be made and do not need to follow any quality control standards, though unofficial standards and common practices can manifest. While the lack of a true standard and quality control would normally cause issues with the quality of drivers and software code potentially, the semantics of RICE can help mitigate these problems.

While distributions of RICE can be marketed, the prospect of free versions of similar systems made from the RICE Kernel keep the price low. Programs can also be created to be marketed or to be given freely to the community.

Current Flavors/Distributions of RICE

Uncooked RICE

Uncooked RICE is a version of the operating system which uses the v0.8.5 RICE Kernel, and can utilize an early version of the “Gohan” or “Bento” Workspace Environment. It is still in development stages as its name implies, but is usable and has a number of programs.

RICE

RICE is a version of the operating system which uses the v1.0.0 RICE Kernel, and can utilize the “Gohan” or “Bento” Workspace Environment. It is the typical distribution supported by its creator and the Sunflower Corporation.

LA+

LA+ ('Laplace') is a distribution of RICE made by the Lagrange Foundation and Lazarus Consortium as a joint-project. It introduces a few advanced versions of existing systems while sharing their source: as well as a number of proprietary applications which can be transplanted from LA+ into any other distribution but the source cannot be viewed. As RICE evolves, new versions of the kernel, core libraries and systems are automatically downloaded and applied in Kernel Updates, ensuring LA+ remains up to date with the latest changes without user intervention.