Introducing TEA OS - A Minimal Operating System Implemented Using the TEA Programming Language

Willrich J. Lutalo*

joewillrich@gmail.com, jwl@nuchwezi.com

June 4, 2026

Abstract

This paper briefly introduces the TEA Operating System (TEA OS); a compact software operating environment runtime built on top of and leveraging the Transforming Executable Alphabet programming language. TEA OS packages a minimal shell, a set of core utilities for doing math, TEA programming and web I/O, as well as a robust, directory-less persistence layer in form of a file system that has a suite of file I/O commands familiar to users of Linux, Unix and MS-DOS. TEA OS intentionally targets userland portability rather than traditional kernel responsibilities, a distinction we clarify and formalize in this paper. We present a preserved verbatim source listing of the current (v1.1.0) system implementation in the TEA language. To illustrate usage and pedagogy, the paper includes an introductory note, the complete source code listing, and a sample interactive session dump showing a user driving the TEA shell. Given that the TEA OS is currently shipped alongside the TEA language as part of the standard TEA package and programs, it is hoped that those evaluating the usability, relevance and power of TEA programming can leverage this example general case to truly come to appreciate the innovation in the language known as the Transforming Executable Alphabet (TEA).

Keywords: Transforming Executable Alphabet, TEA, Applying TEA, Minimal Operating System, Text-Processing; Software Operating Environments, Preprint, Source Code.

1 Introduction

Considering all the potential ways a [general-purpose] programming language might be utilized, realizing an operating system is among the most critically

*Inventor of the TEA language, also currently a volunteering & Independent Principal Investigator at Nuchwezi Research — https://nuchwezi.com

important ends one might consider [1]. Having already finished introducing the world to the Transforming Executable Alphabet (TEA) language in [2], it is indeed necessary to go ahead and likewise formally introduce some of the special cases it has been well applied to.

The TEA Operating System (TEA OS), also sometimes referred to as "TOS", is a software operating environment (SOE) [3] meant to allow for more than just writing, storing and running [TEA] programs; it allows for the writing and evaluation of mathematical expressions; fetching, saving, processing, and presentation of web-based/network-accessible resources; as well as ready access-to a user-aware, extensible and minimal, self-documenting shell known as the TEA OS Shell (perhaps better we start calling it "TOSH" to keep with traditional shell-naming conventions).

It shall be worth noting that TEA OS was designed with a different set of assumptions than conventional operating systems: rather than assuming privileged hardware access and low-level languages, TEA OS assumes a TEA runtime baseline and prioritizes minimalism and portability. This design enables TEA OS to be expressed compactly and to run in environments such as web browsers without a virtual machine; it also means that TEA OS should be read as a userland runtime and service stack unless explicitly extended with kernel-level primitives.

The complete design, justification, specification and demonstrations of the current TEA OS implementation are already well covered in the relevant chapter in the TEA language manual [1]. We shall not waste time and space reproducing those bits here then. However, it should be worth noting that, from the onset, the design of TEA OS was without doubt inspired by its inventor's awareness of the ideas underlying popular traditional mainstream operating systems such as Unix, Linux and the currently almost defunct MS-DOS.

Next, we are going to consider the initial, complete specification of the TEA OS in **Section 2** as first laid down in [1]. We shall consider the TEA OS architecture in **Section 3**, and shall discuss how TEA OS relates to, differs from traditional operating system kernels in **Section 4**. **Section 5** presents the verbatim form of the complete source code of **TEA OS v1.1.0** also which can be found in its original, but also up-to-date form via `https://bit.ly/run_teaos`, and then we shall reproduce a note originally introducing v1.1.0 of the OS in **Section 6**, as well as a verbatim session dump — exported from the associated TOSH in **Section 7**. We then conclude in **Section 8** with key observations made as well as what the potential future directions are.

2 The TOS Specification

Without the sanction of some working-group or committee, we have thought-up and drafted the following minimal specification as the essential basis of the **TEA Operating System**:

TOS Specification:

The TEA Operating System is specified based on 4 core capabilities:

1. File Input/Output Service
2. [TEA] Programming Service
3. Basic Utility Service
4. Help/Documentation Service
5. UI/Interface Service

Moreover, and perhaps very worthwhile to note, we learn from one of the reference books on Linux — a globally popular, and dominant free operating system — that besides the core (or rather "kernel") of an operating system, it is quite important to be clear on what services and utilities render the operating system usable. See what Ellen Siever and Stephen Spainhour et al. have to say about this:

> `The Linux kernel itself was originally designed by Linus Torvalds at the University of Helsinki in Finland and later developed through collaboration with countless volunteers worldwide. By "kernel," we mean the core of the operating system itself---not the applications (such as the compiler, shells, and so forth) that run on it. Today, the term "Linux" is often used to mean the kernel as well as the applications and complete system environment.`
>
> —**Ellen Siever and Stephen Spainhour et al.**, *Linux in a Nutshell: A Desktop Quick Reference*, 2000[4]

And with that clarification out of the way, we then embark on specifying the details of our operating system — and somewhat, to again reflect the manner and approach of [4] in clarifying what makes Linux tick as well as how it is used, we shall especially approach the specification as though we were laying down a specification of the interaction interface that users of TOS rely upon to use the operating system for arbitrary tasks. The four categories or capabilities laid down in the mini-specification above shall guide us...

1. **File System:** create, name, rename, copy, lookup, preview, and delete file objects.
   —**[Basic FILE I/O Commands Interface]:**

- `create FILENAME` ← creates blank file with name FILENAME. Valid FILENAME should consist of ONLY letters a-zA-Z, numbers 0-9, underscore _ and/or a dot ..
- `init FILENAME` ← create a file using the given name and the currently available input/last-output (or the **print buffer**) or blank if none.
- `write FILENAME` ← write the contents of the current available input/last-output (or the **print buffer**) or blank if none, to a file with the specified name. If the file already exists, essentially overwrites its contents.
- `exists FILENAME` ← Show whether or not a file named FILENAME exists.
- `(rename | rn | mv | move) FILENAME NEWNAME` ← Change the name of an existing file from FILENAME to NEWNAME. WILL ALSO update the registered [user] programs list if FILENAME was a registered program.
- `(copy|clone) FILE1 FILE2` ← Copies contents of FILENAME to FILE2 creating the destination if it doesn't exist.
- `find PATTERN` ← Display list of ONLY files matching regular expression PATTERN.
- `(show | cat | read) FILENAME` ← Preview the contents of given file.
- `export FILENAME` ← Exit the system, outputting the contents of given file.
- `(del | delete | rm) PATTERN` ← Remove from storage, any and all files whose name matches PATTERN. CAN NOT DELETE registered user program files though.
- `ls | list | dir | files` ← Display list of ALL available files.
- `(ls | list | dir | files)` PATTERN ← Display list of ONLY files matching regular expression PATTERN.

2. **Programming Interface:** a command-line interface to find, run, or modify user programs or find and run system programs and commands.
   —[**Basic Programming Commands Interface**]:

   - `(runnable | register | add-program) FILENAME` ← Recognize FILENAME object as an executable [user] program or command. FILENAME must already be an existing TEA file.
   - `programs [PATTERN]` ← Show list of all available registered [TEA] user programs by their registered names. With PATTERN specified, ONLY show those programs whose names match regular expression PATTERN.

- `(isrunnable | canrun) FILENAME` ← Show whether or not a file named FILENAME can be executed as a [TEA] program.
- `(run | exec | execute) FILENAME` ← Execute the given file record as a [TEA] program.
- `(del-prog | delete-program | unregister) FILENAME` ← Revokes status of FILENAME as an executable user program (doesn't actually delete the underlying file object though).
- `(update | update-program) FILE1 FILE2` ← Replace/override original program named FILE1 with contents of FILE2 (updates a program's code [in FILE1] preserving its existing name). Both files FILE1 and FILE2 must actually exist for this to work, but only FILE1 needs to be a registered program.
- `PROGRAM` ← Executes PROGRAM as a system command if it exists — first, if it is any of the standard commands in TEA OS, otherwise if it is a valid filename "PROGRAM" that is already recognized/registered as a user-created program — this later case similar to calling "run PROGRAM", otherwise shows error message command-not-found.
- `(help | man | whatis | info | about) PROGRAM` ← displays help info about a given system command or user program.

3. **Basic Utilities:** access to basic useful commands such as being able to interact with the world outside TOS as well as do non-programming work from within TOS.
   —[**Basic UTILITY Commands Interface**]:

   - `(clear | reset)` ← Resets and clears the current TEA OS session log since login.
   - `date` ← return the current date.
   - `echo` MESSAGE ← Simply print MESSAGE as is.
   - `exit` ← ends the TOS session returning nothing.
   - `export` ← Take the output of the last command or rather, current contents of the output buffer and export that outside the operating system (also terminates the operating system session).
   - `(fetch | web) URL` ← fetch and output contents of the specified [online/network] resource.
   - `fetch URL FILENAME` ← fetch and save contents at URL into file record named FILENAME.
   - `logout` ← Resets username and system logs and ends the TOS session returning nothing.
   - `math EXPRESSION` ← execute basic mathematics on the command-line as specified in EXPRESSION.

- `print` ← Show or preview current output buffer contents (usually, output of last command). After clearing log, but also after exit/system shutdown/login also shows final output before the log was reset.
- `tea CODE..` ← take everything following the keyword "tea" as a TEA program, run and output the result: It's a simple way to write and run basic TEA programs (TEA-one-liners) in TOS.
- `time` ← return the current time.
- `timestamp` ← return the current timestamp.
- `username NAME` ← Overrides current username with specified value.
- `(whoami|name)` ← Display the NAME of currently active USER.

4. **Help System:** a means to locate and read basic documentation about programs, system commands and the operating system.
   —**[Basic HELP Commands Interface]:**

   - `about` ← Displays basic info about the currently running version of the TEA Operating System.
   - `help | man | whatis | info` ← present menu for user to access help about all available [essential] commands based on category.
   - `(help | man | whatis | info | about) NAME` ← displays help info about a given system command or user program specified in the value NAME.
   - `help-file` ← display all available commands in the FILE I/O category.
   - `help-help` ← display help on how to use the HELP facility.
   - `help-prog` ← display all available commands in the PROGRAMMING category.
   - `help-ui` ← Display all available commands in the USER INTERFACE category or sub-system.
   - `help-util` ← display all available commands in the UTILITIES category.

After working on and testing the initial prototypes of the TEA OS, it was established that we shall need some means of controlling and configuring the user-interface mechanisms, and thus the following extra sub-system specification:

1. **INTERFACE Utilities:** access commands that can alter the user-experience and how certain user-interface aspects of the operating system work within TOS.
   —**[Basic UI/UX Commands Interface]:**

- `uimini` ← Configure TEA OS interface to render using mini-terminal mode. Setting persists across logins until changed. This mode is suitable for environments such as using TOS on some Desktop Computer Systems.
- `uilog` ← Configure TEA OS interface to render using long/log-terminal mode. Setting persists across logins until changed. This mode is suitable for environments such as using TOS on the TEA MOBILE IDE.

And thus, the plan is now officially underway... to manifest an new operating system based on TEA!

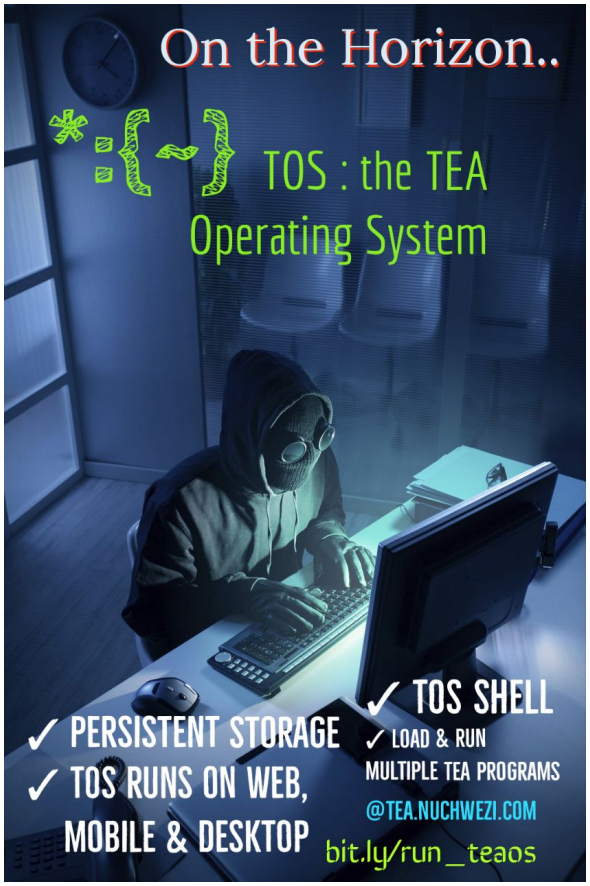
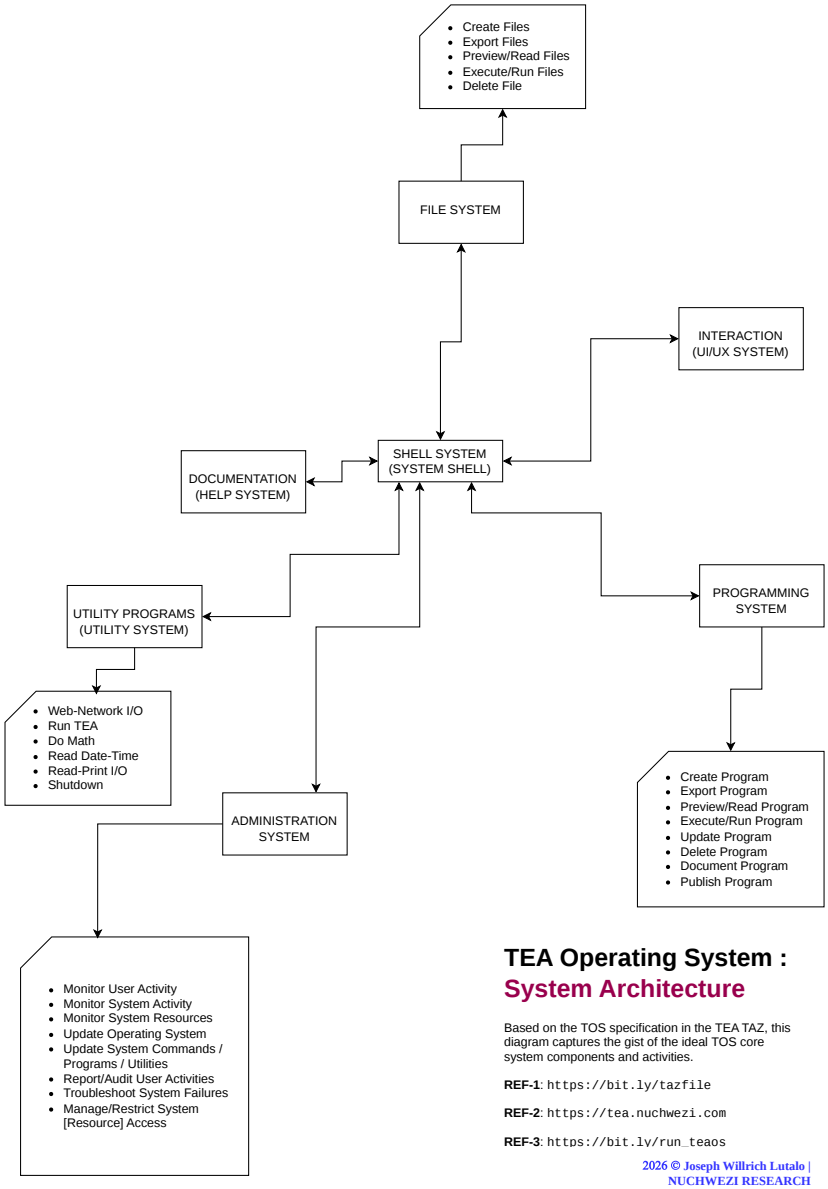


Figure 1: **A Hacker's Dream Come True?** The proposal to bring to life a **TEA Operating System**!

The link depicted in **Figure 1** being the first to have been officially shared with members of the TEA community concerning the original TEA OS pro-

posal, we had it point, not to the running instance of the TOS, but rather, to the potential staging source-file for the reference implementation of the operating system. And thus, those zealously interested in following what the latest working version of the single-file TEA OS code looks like — a program that one should be able to readily load into either the TEA WEB IDE or verify on any system with `tttt` package installed, and which they can then readily run, shall be found at the link: `https://bit.ly/run_teaos`

3 TEA OS Architecture



4 Contrasting TEA OS Architecture with Traditional Kernel-Centric Operating Systems

This section situates the TEA OS design presented in **Section 3** within the broader taxonomy of operating systems[5] and explains, in concrete technical terms, how TEA OS differs from a traditional kernel-centric implementation. The TEA OS was intentionally designed as a **minimal, portable runtime and service stack** — the TEA OS source (which we shall come to in), explicitly mentions the inventor's intentions as:

Table 1: TEA OS's inventor concerning the intended TEA OS core design

> *a true general purpose, lightweight, minimal but powerful operating system that can essentially just fit inside a single string*

It should be worth noting that final aspect... "that fits inside a string", because it shall without doubt explain why and how it is that the reference implementation of the TEA operating system is currently being shipped alongside other *standard* TEA programs[2] that essentially are loadable into the TEA WEB IDE via a mere JSON payload!

Further, note that it was originally intended that the OS should be able to run wherever the TEA runtime baseline is available (including directly in web browsers as already mentioned). That design goal drives a set of architectural tradeoffs that depart from the assumptions and responsibilities normally associated with an OS kernel.

4.1 High-level Summary of Intent

TEA OS is a compact, self-contained runtime (also what, in the terminology of Lutalo et al. we might technically refer to as a *software operating environment* [3]) that exposes five core capabilities: a **file system**, a **utility system**, a **help system**, a **programming system**, and a **UI system**. Interaction is primarily through a shell that supports the TEA language interpreter(s) and user-extensible commands. The project deliberately targets portability and minimalism: TEA OS assumes the presence of a TEA runtime rather than privileged hardware access, and it is implemented in a high-level language designed to be more compact and portable than mainstream scripting or programming languages [1][1]. As a consequence, TEA OS can execute in environments where traditional kernels cannot (for example, directly inside a web browser without a virtual machine).

[1] Also refer to **Appendix A** on "Fundamentals of Programming via TEA" as laid down in [1].

4.2 Core Differences: Responsibilities and Assumptions

The following table summarizes the most relevant technical attributes and contrasts TEA OS against the canonical responsibilities of a traditional kernel. Each cell is a concise statement of whether the capability is typically a kernel responsibility and whether TEA OS implements it in its current form.

Attribute	Typical Kernel	TEA OS (current)
Hardware control and drivers	Direct device drivers; interrupt handling	Not implemented; relies on host runtime I/O
CPU scheduling and multitasking	Preemptive scheduling; context switching	Cooperative or single-threaded within TEA runtime
Memory management and protection	Virtual memory; isolation; MMU use	Managed by host runtime; no low-level protection
System call / syscall interface	Privileged syscall boundary	High-level API exposed by TEA runtime
Boot and initialization	Bootloader; kernel startup	TEA program initialization within host runtime
File system	Often kernel or kernel-assisted	Implemented as a TEA file subsystem (userland)
Security and privilege separation	Kernel enforces privileges	Relies on host sandboxing and TEA runtime
Extensibility (user commands)	Usually userland shells	Native: shell + TEA interpreters for extensions
Portability target	Hardware/ISA specific	Portable across TEA runtimes and browsers

Table 2: Checklist contrasting typical kernel responsibilities with TEA OS design.

4.3 Architectural Implications

Privilege and Trust Model. Traditional kernels operate in a privileged CPU mode and mediate access to hardware and shared resources [5, 6]. TEA OS, by contrast, is designed to run in an unprivileged environment provided by the TEA runtime [1]. This shifts the trust boundary: TEA OS trusts the host runtime (and the host environment, e.g., the browser) to provide isolation, I/O, and resource accounting. The practical implication is that TEA OS cannot, in its present form, enforce hardware-level isolation or implement mandatory

access control independent of the host.

Portability and Minimality. Because TEA OS targets a high-level runtime, it attains portability across platforms that implement the TEA baseline. This is a deliberate tradeoff: TEA OS sacrifices direct hardware control and low-level performance optimizations in exchange for extreme portability and compactness. The design goal of "fitting inside a string" and running without a VM is a distinguishing feature that enables deployment scenarios (e.g., in-browser shells, embedded documentation with runnable OS examples) that traditional kernels cannot match.

Service Placement and Layering. TEA OS places many services that would traditionally be split between kernel and userland entirely in the TEA userland layer (file system abstraction, utilities, help, programming environment, UI). This simplifies the kernel model (there is effectively no kernel in the traditional sense) but increases the responsibility of the TEA runtime and the host to provide secure, deterministic primitives.

4.4 When can TEA OS be called a Kernel?

The term *kernel* is meaningful only relative to a set of responsibilities [7]. If we wished to claim that TEA OS is a kernel, then we would be "conventionally" tasked to explicitly document or explain which of the canonical kernel responsibilities the TEA OS implements. Below is a practical checklist of items that, if implemented and documented, would justify calling TEA OS a **kernel** in the conventional sense:

- **Boot and initialization:** a documented boot sequence that initializes hardware or a hardware abstraction layer.
- **Device drivers and I/O:** drivers or a driver framework that directly manage devices or a well-specified hardware abstraction layer.
- **Interrupt handling:** mechanisms to receive and dispatch hardware interrupts or their equivalents.
- **Memory management:** support for memory allocation with protection boundaries (virtual memory or equivalent isolation).
- **Process scheduling:** preemptive multitasking or a scheduler that arbitrates CPU among isolated processes/threads.
- **System call interface:** a privileged boundary and syscall ABI for user programs to request kernel services.
- **Privilege separation and security:** enforcement of privilege levels and resource quotas independent of the host runtime.

- **Persistence and storage drivers:** direct control or well-specified management of persistent storage devices.

If TEA OS implements a substantial subset of these items (and documents the mechanisms and guarantees), it would then be reasonable to describe it as a kernel or as a kernel-like runtime [7]. However, since it currently does not, we shall currently be satisfied with the more accurate description of the TEA OS as a **minimal OS runtime and service stack**, as a "technology platform", or as a userland **software operating environment** built on top of the TEA runtime.

5 TEA OS v1.1.0 source

```
#!/usr/bin/tttt -fc
I!:{oooooooo_________________
___oo_____ooooo___ooooo__
___oo____oo____o_oo___oo_
___oo____ooooooo_oo___oo_
___oo____oo______oo___oo_
___oo_____ooooo___oooo_o_
_________________________}
x!:{
TEA OS v1.1.0 is Ready...}
v:vBanner | #i:

###| The above is the "original" bootscreen
###| The rest of the story follows..

*:{==| PREAMBLE |==}
L:lPREAMBLE_START

Y:vBanner #| V@:dvLOG
V@:dvLOG_start

#Of course, this is TEA programming
We can write verbatim documentation and
comments in the most natural of ways without
worry or concern. We sometimes shall
do just that.

V@:dvABOUT_TOS:{
#---[ABOUT TEA OS]

NAME: TEA Operating System
(also "TEA OS" or just "TOS")

ARCHITECT: J. Willrich Lutalo <jwl@nuchwezi.com>

```

```
SINCE: 9th APRIL, 2026

INTENT: To manifest a true general purpose, lightweight,
    minimal but powerful operating system that can
    essentially just fit inside a single string; to see what
    the limits of the mathematics of TRANSFORMATICS might be,
     and what we surely could accomplish with its proper use
    via modern portable computers.

ABOUT: TEA OS is implemented using the Transforming
    Executable Alphabet (TEA) programming language. It is
    based on a specification first laid out in the TEA TAZ.
    It is a project originating from UGANDA.

PROJECT HOME: Nuchwezi Research | https://tea.nuchwezi.com}

#--[GLOBAL CONSTANTS]
V@:dvOSName:{TEA OS}
V@:dvOSName_short:{TOS}
V@:dvOSVersion:{1.1.0}

V:vSLOGAN:{TEA OS! T can fit in a string!}
I!:{____________________}|V:vDELIM|V@:dvDELIM
I!:{
=o==o==o==o==o==o==o=
}|V:vDELIM2|V@:dvDELIM2


#--[GLOBAL VARIABLES]
V:vUSERNAME:{user} # not used that much!


#--[DATABASE VARIABLES]
#Y@:dvLOG|i:{}|V@:dvLOG #all activity
V@:dvLastPS:{}
V@:dvLastCommand:{}
V@:dvLastOutput:{}

#TEA-DATABASE powered FILE-SYSTEM register
#{F1|F2|...|FN} s.t NAME(Fi)=ID(Fi)
Y@:dvFileList|I:{.}|V@:dvFileList

#TEA-DATABASE powered USER-PROGRAMS-SYSTEM register
#{F1|F2|...|FN} s.t NAME(Fi)=ID(Fi)
Y@:dvProgramsList|I:{run}|V@:dvProgramsList

####| INTERFACE CONFIG VARIABLE
# 0 forces original terminal mode: fit for TEA mobile IDE
# 1 is mini-terminal mode: fit for CLI and other WEB
# user can change this via commands:
```

```
# uimini -> 1
# uilog -> 0
# will default to mini if not already set
Y@:dvUIMODE_MINI|I:{1}|V@:dvUIMODE_MINI

#--[SOME FUNCTIONS]

#for presenting pretty message prompt..
# in vUIMODE_MINI=0
V:fPROMPT_DEFAULT:"
# FIRST: IMPORTANTLY...
v:vMSG #stores AI at invocation
y@:dvDELIM|v:vD2
y:vMSG|x!:{ }|v:vMSG #affix space at prompt end
i!:{} | g*:{
}:vD2:vMSG
v:vMSG
Y@:dfLogAI|v:f # a stealth import :)
y:vMSG|e*:f
#then show log+latest prompt
Y@:dvLOG
i: #returns input..
"

Y:fPROMPT_DEFAULT|V@:dfPROMPT_DEFAULT

#for presenting pretty message prompt..
# in vUIMODE_MINI=1
V:fPROMPT:"
# FIRST: IMPORTANTLY...
v:vMSG #stores AI at invocation

#check if we should instead be using default ui
y@:dvUIMODE_MINI | f!:^0$:lN_UIMODE_MINI
y@:dfPROMPT_DEFAULT|v:fPROMPT_DEFAULT
y:vMSG | E*:fPROMPT_DEFAULT
q!: # no need to proceed..
l:lN_UIMODE_MINI

#THEN imports...
y@:dfLastIO|v:fLastIO
y@:dfMakePS|v:fMakePS

y@:dvDELIM|v:vD2
y:vMSG|x!:{ }|v:vMSG #affix space at prompt end
i!:{} | g*:{
}:vD2:vMSG
v:vMSG
Y@:dfLogAI|v:f # a stealth import :)
y:vMSG|e*:f
```

```
#then show log+latest prompt
#Y@:dvLOG
E*:fMakePS|v:vNEXTPS
E*:fLastIO #fetch last I/O
x!:{
> }|#x*!:vNEXTPS
i: #returns input..
"

Y:fPROMPT|V@:dfPROMPT

#login function
V:fLogin:"
#import some functions from database...
y@:dfPROMPT_DEFAULT|v:fPROMPT
y@:dfLogAI|v:fLogAI
y@:dfLogO|v:fLogO
y@:dfGreetUser|v:fGreetUser

# show OS banner
y@:dvLOG_start | v:vSCREEN
e*:fLogAI

#greet user via stored name..
y!@:dvUSERNAME | f!:^0$:lGREET_USER
#Username not yet set, prompt and store
l:lSET_USERNAME
I!:{
Please set a Username:}|v:vPromptMSG
e*:fLogAI
y:vSCREEN | x*!:vPromptMSG | i:
t!.:|v:|v!:|f:^0$:lSET_USERNAME
y:|V@:dvUSERNAME # store in database
e*:fLogAI

l:lGREET_USER
e*:fGreetUser|v:vGreetings
e*:fLogO | e*:fPROMPT
"

#load username and greet
V:fGreetUser:"
y@:dvUSERNAME
x:{
Hello, }|x!:{. Welcome to TOS!}
"

Y:fGreetUser|V@:dfGreetUser

#for os name..
```

```
V:fOSHandle:"
y@:dvOSName | v:vN
y@:dvOSVersion | v:vV
g*:{ v}:vN:vV
"

Y:fOSHandle|V@:dfOSHandle

#for default prompt..
V:fMakePS:"
z.:TIME |v:vT
y@:dvOSName_short | v:vPWD
v:vD:{>}
g*:{/}:vPWD:vT:vD
"

Y:fMakePS|V@:dfMakePS

#log last cmd+O
V:fLogIO:"
#load+execute function from database!
y@:dfLastIO|e:
v:vLastIO
y@:dvLOG | v:vLOG
g*:{
}:vLOG:vLastIO | v@:dvLOG
"

#suffix AI to log
V:fLogAI:"
v:vAI
y@:dvLOG | v:vLOG
g*:{
}:vLOG:vAI | v@:dvLOG
#return AI
y:vAI
"

#also make it a db function
Y:fLogAI|v@:dfLogAI

#store AI as last-O..
V:fLogO:"
v@:dvLastOutput
v@:dvDATABUFFER #also stash into special data buffer
Y@:dvLastOutput
"

#fetch last I/O..
V:fLastIO:"
```

```
y@:dvLastPS | v:vPS
y@:dvLastCommand | v:vC
y@:dvLastOutput | v:vO
g*:{}:vPS:vC|v:vCC
g*:{
}:vCC:vO
"
#store that function in the database too (so we can later
    use it in context-unaware functions).
y:fLastIO|v@:dfLastIO

L:lPREAMBLE_END
#:{=================}

*:{==|UTILITY SYSTEM |==}
L:lSYS_UTILITY_START

#--[UTILITY Functions]

#fix URL
V:fFixURL:"
v:vURL|z:
f:{^https?://}:lFINE
y:vURL|x:{https://}
v:vURL
l:lFINE
y:vURL
"

#reset+clear log
V:fResetLog:"
y@:dvLastOutput | v@:dvLastOutput_backup
c!@:dvLOG
c*:vLOG
C!@:dvLastOutput
y@:dfGreetUser|E:|v:vGreetings
y@:dvLOG_start
x*!:vGreetings
"

# make available for [db] import..
Y:fResetLog|V@:dfResetLog


V:fGetLatestOutputBuffer:"
y@:dvLastOutput_backup
f!:^$:lN_USED_BACKUP
y@:dvLastOutput
l:lN_USED_BACKUP
c@:dvLastOutput_backup
```

```
277 "
278
279 Y:fGetLatestOutputBuffer|V@:dfGetLatestOutputBuffer
280
281 L:lSYS_UTILITY_END
282 #:{=================}
283
284 *:{==|HELP SYSTEM |==}
285 L:lSYS_HELP_START
286
287 #--[HELP/DOC Functions]
288
289 #present USER INTERFACE help-category commands
290 V:fGetUIMenu:"
291 y@:dfOSHandle|e:|x!:{ INTERFACE System Commands:
292 uimini
293 uilog}
294 v:vMSG
295 y@:dvDELIM2|v:vD
296 g*:{}:vD:vMSG:vD
297 "
298
299 #present FILE help-category commands
300 V:fGetFileCMDMenu:"
301 y@:dfOSHandle|e:|x!:{ FILE System Commands:
302 create FILENAME
303 init FILENAME
304 write FILENAME
305 (rename | rn | mv | move) FILENAME NEWNAME
306 (copy|clone) FILE1 FILE2
307 exists FILENAME
308 find PATTERN
309 (show | cat | read) FILENAME
310 export FILENAME
311 (del | delete | rm) FILENAME
312 (ls | list | dir | files) PATTERN}
313 v:vMSG
314 y@:dvDELIM2|v:vD
315 g*:{}:vD:vMSG:vD
316 "
317
318 #present PROGRAMMING help-category commands
319 V:fGetProgrammingMenu:"
320 y@:dfOSHandle|e:|x!:{ PROGRAMMING System Commands:
321 (runnable | register | add-program) FILENAME
322 (isrunnable | canrun) FILENAME
323 (run | exec | execute) FILENAME
324 (del-prog | delete-program | unregister) FILENAME
325 (update | update-program) FILENAME
326 PROGRAM
```

```
(help | man | whatis | info | about) PROGRAM}
v:vMSG
y@:dvDELIM2|v:vD
g*:{}:vD:vMSG:vD
"

#present UTILITIES help-category commands
V:fGetUtilityMenu:"
y@:dfOSHandle|e:|x!:{ UTILITY System Commands:
clear | reset
date
echo MESSAGE
exit
export
(fetch | web) [URL] [FILENAME]
logout
math EXPRESSION
print
tea CODE
time
timestamp
username NAME
whoami | name}
v:vMSG
y@:dvDELIM2|v:vD
g*:{}:vD:vMSG:vD
"

#present main help-category commands
V:fGetHelpMenu:"
y@:dfOSHandle|e:|x!:{ HELP System Commands:
about
(help | man | whatis | info | about) [NAME]
help-file
help-help
help-prog
help-ui
help-util}
v:vMSG
y@:dvDELIM2|v:vD
g*:{}:vD:vMSG:vD
"

#present main about os info
V:fGetAboutTOS:"
y@:dfOSHandle|e:|x:{ABOUT: }
v:vMSG
y@:dvABOUT_TOS|x*:vMSG|v:vMSG
y@:dvDELIM2|v:vD
g*:{}:vD:vMSG:vD
```

```
"

#present help-on a given topic
V:fGetHelpOn:"
#asumes topic is passed via AI
v:vHelpTOPIC

#some imports...
y@:dfGetFilenameProtocolMSG|v:fGetFilenameProtocolMSG
E*:fGetFilenameProtocolMSG | v:vFILENAME_PROTOCOL

#set default..
r@:{That help topic [#vHelpTOPIC#] is not understood or
    supported. Try the 'help' command to see what help
    commands there are. For example 'help whatis'}|v:vMSG

y:vHelpTOPIC

#> help about
f!:^about$:lN_h0
v:vMSG:{about <~ Displays basic info about the currently
    running version of the TEA Operating System.}
j:lh1_present
l:lN_h0

#> help whatis
f!:^whatis$:lN_h1
v:vMSG:{whatis NAME <~ Displays help info about a given
    system command or user program.}
j:lh1_present
l:lN_h1

#> help man
f!:^man$:lN_h1b
v:vMSG:{man NAME <~ Displays help info about a given system
    command or user program. It is just another alias for '
    whatis' or 'help'}
j:lh1_present
l:lN_h1b

#> help help
f!:^help$:lN_h2
v:vMSG:{Present menu for user to access help about all
    available [essential] commands based on category.}
j:lh1_present
l:lN_h2

f!:^help-file$:lN_h3
v:vMSG:{Display all available commands in the FILE I/O
    category or sub-system.}
```

```
419 j:lh1_present
420 l:lN_h3
421
422 f!:^help-prog$:lN_h4
423 v:vMSG:{Display all available commands in the PROGRAMMING
        category or sub-system.}
424 j:lh1_present
425 l:lN_h4
426
427 f!:^help-util$:lN_h5
428 v:vMSG:{Display all available commands in the UTILITIES
        category or sub-system.}
429 j:lh1_present
430 l:lN_h5
431
432 f!:^help-help$:lN_h6
433 v:vMSG:{Display help on how to use the HELP facility or sub-
        system.}
434 j:lh1_present
435 l:lN_h6
436
437 f!:{^(clear|reset)$}:lN_hu0
438 v:vCMD
439 r@:{#vCMD#  <~ Resets and clears the current TEA OS session
        log since login.} | v:vMSG
440 j:lh1_present
441 l:lN_hu0
442
443 f!:{^(fetch|web)$}:lN_hu1
444 v:vCMD
445 r@:{#vCMD# URL <~ fetch and output contents of the specified
         [online/network] resource.
446
447 #vCMD# URL FILENAME <~ fetch and save contents at URL into
        file record named FILENAME. #vFILENAME_PROTOCOL#} | v:
        vMSG
448 j:lh1_present
449 l:lN_hu1
450
451 f!:^tea$:lN_hu2
452 v:vMSG:{tea CODE.. <~ take everything following the keyword
        'tea' as a TEA program, run and output the result: It's a
         simple way to write and run basic TEA programs (TEA-one-
        liners) in TOS.}
453 j:lh1_present
454 l:lN_hu2
455
456 f!:^math$:lN_hu3
457 v:vMSG:{math EXPRESSION <~ execute basic mathematics on the
        command-line as specified in EXPRESSION.}
```

```
458 j:lh1_present
459 l:lN_hu3
460
461 f!:^date$:lN_hu4
462 v:vMSG:{date <~ Show the current date.}
463 j:lh1_present
464 l:lN_hu4
465
466 f!:^time$:lN_hu5
467 v:vMSG:{time <~ Show the current time.}
468 j:lh1_present
469 l:lN_hu5
470
471 f!:^timestamp$:lN_hu6
472 v:vMSG:{timestamp <~ Show the current timestamp.}
473 j:lh1_present
474 l:lN_hu6
475
476 f!:^print$:lN_hu7
477 v:vMSG:{print <~ Show or preview current output buffer
        contents (usually, output of last command). After
        clearing log, but also after exit/system shutdown/login
        also shows final output before the log was reset.}
478 j:lh1_present
479 l:lN_hu7
480
481 f!:^export$:lN_hu8
482 r@:{export <~ Take the output of the last command or rather,
         current contents of the output buffer and export that
        outside the operating system (also terminates the
        operating system session).
483
484 export FILENAME <~ Exit the system, outputting the contents
        of given file. #vFILENAME_PROTOCOL#} | v:vMSG
485 j:lh1_present
486 l:lN_hu8
487
488 f!:^exit$:lN_hu9
489 v:vMSG:{exit <~ Ends the TOS session returning nothing.}
490 j:lh1_present
491 l:lN_hu9
492
493 f!:^logout$:lN_hu10
494 v:vMSG:{logout <~ Resets username and system logs and ends
        the TOS session returning nothing.}
495 j:lh1_present
496 l:lN_hu10
497
498 f!:^username$:lN_hu11
```

```
499 v:vMSG:{username NAME <~ Overrides current username with
        specified value.}
500 j:lh1_present
501 l:lN_hu11
502
503 f!:^help-ui$:lN_h12
504 v:vMSG:{Display all available commands in the USER INTERFACE
         category or sub-system.}
505 j:lh1_present
506 l:lN_h12
507
508 f!:^uimini$:lN_h13
509 v:vMSG:{uimini <~ Configure TEA OS interface to render using
         mini-terminal mode. Setting persists across logins until
         changed. This mode is suitable for environments such as
        using TOS on some Desktop Computer Systems.}
510 j:lh1_present
511 l:lN_h13
512
513 f!:^uilog$:lN_h14
514 v:vMSG:{uilog <~ Configure TEA OS interface to render using
        long/log-terminal mode. Setting persists across logins
        until changed. This mode is suitable for environments
        such as using TOS on the TEA MOBILE IDE.}
515 j:lh1_present
516 l:lN_h14
517
518 f!:^whoami$:lN_h15
519 v:vCMD
520 r@:{#vCMD# <~ Display the NAME of currently active USER.} |
        v:vMSG
521 j:lh1_present
522 l:lN_h15
523
524 f!:^create$:lN_h16
525 r@:{create FILENAME <~ Creates blank file with name FILENAME
        . #vFILENAME_PROTOCOL#}|v:vMSG
526 j:lh1_present
527 l:lN_h16
528
529 f!:{^(ls|list|dir|files)$}:lN_h17
530 v:vCMD
531 r@:{#vCMD# <~ Display list of ALL available files.
532
533 #vCMD# PATTERN <~ Display list of ONLY files matching
        regular expression PATTERN.}
534 | v:vMSG
535 j:lh1_present
536 l:lN_h17
537
```

```
f!:^init$:lN_h18
r@:{init FILENAME <~ Create a file using the given name and
    the currently available input/last-output (or the PRINT
    buffer) or blank if none. #vFILENAME_PROTOCOL#}|v:vMSG
j:lh1_present
l:lN_h18

f!:^write$:lN_h19
r@:{write FILENAME <~ Write the contents of the current
    available input/last-output (or the PRINT buffer) or
    blank if none, to a file with the specified name. If the
    file already exists, essentially overwrites its contents.
     #vFILENAME_PROTOCOL#} | v:vMSG
j:lh1_present
l:lN_h19

f!:{^(show|cat|read)$}:lN_h20
v:vCMD
r@:{#vCMD# FILENAME <~ Preview the contents of given file. #
    vFILENAME_PROTOCOL#} | v:vMSG
j:lh1_present
l:lN_h20

f!:^echo$:lN_h21
v:vMSG:{echo MESSAGE <~ Simply print MESSAGE as is.}
j:lh1_present
l:lN_h21

f!:{^(find)$}:lN_h22
v:vCMD
r@:{#vCMD# PATTERN <~ Display list of ONLY files matching
    regular expression PATTERN.}
| v:vMSG
j:lh1_present
l:lN_h22

f!:{^(del|delete|rm)$}:lN_h23
v:vCMD
r@:{#vCMD# PATTERN <~ Remove from storage, any and all files
     whose name matches PATTERN. CAN NOT DELETE registered
    user program files though.} | v:vMSG
j:lh1_present
l:lN_h23

f!:{^(rename|rn|mv|move)$}:lN_h24
v:vCMD
r@:{#vCMD# FILENAME NEWNAME <~ Change the name of an
    existing file from FILENAME to NEWNAME. WILL ALSO update
    the registered [user] programs list if FILENAME was a
    registered program. #vFILENAME_PROTOCOL#} | v:vMSG
```

```
575 j:lh1_present
576 l:lN_h24
577
578 f!:{^(exists)$}:lN_h25
579 v:vCMD
580 r@:{#vCMD# FILENAME <~ Show whether or not a file named
        FILENAME exists. #vFILENAME_PROTOCOL#} | v:vMSG
581 j:lh1_present
582 l:lN_h25
583
584 f!:{^(copy|clone)$}:lN_h26
585 v:vCMD
586 r@:{#vCMD# FILENAME FILE2 <~ Copies contents of FILENAME to
        FILE2 creating the destination if it doesn't exist. #
        vFILENAME_PROTOCOL#} | v:vMSG
587 j:lh1_present
588 l:lN_h26
589
590 f!:{^(runnable|register|add-program)$}:lN_h27
591 v:vCMD
592 r@:{#vCMD# FILENAME <~ Recognize FILENAME object as an
        executable [user] program or command. FILENAME must
        already be an existing TEA file. #vFILENAME_PROTOCOL#} |
        v:vMSG
593 j:lh1_present
594 l:lN_h27
595
596 f!:{^programs$}:lN_h28
597 v:vCMD
598 r@:{#vCMD# <~ Show list of all available registered [TEA]
        user programs by their registered names.
599
600 #vCMD# PATTERN <~ With PATTERN specified, ONLY show those
        programs whose names match regular expression PATTERN.}
601 | v:vMSG
602 j:lh1_present
603 l:lN_h28
604
605 f!:{^(run|exec|execute)$}:lN_h29
606 v:vCMD
607 r@:{#vCMD# FILENAME <~ Execute the given file record as a [
        TEA] program. #vFILENAME_PROTOCOL#} | v:vMSG
608 j:lh1_present
609 l:lN_h29
610
611 f!:{^(isrunnable|canrun)$}:lN_h30
612 v:vCMD
613 r@:{#vCMD# FILENAME <~ Show whether or not a file named
        FILENAME can be executed as a [TEA] program. #
        vFILENAME_PROTOCOL#} | v:vMSG
```

```
614 j:lh1_present
615 l:lN_h30
616
617 f!:{^(del-prog|delete-program|unregister)$}:lN_h31
618 v:vCMD
619 r@:{#vCMD# FILENAME <~ Revokes status of FILENAME as an
        executable user program (doesn't actually delete the
        underlying file object though). #vFILENAME_PROTOCOL#} | v
        :vMSG
620 j:lh1_present
621 l:lN_h31
622
623
624 f!:{^(update|update-program)$}:lN_h32
625 v:vCMD
626 r@:{#vCMD# PROGRAMFILE FILENAME <~ Replace/override original
         program named PROGRAMFILE with contents of FILENAME (
        updates a program's code [in PROGRAMFILE] preserving its
        existing name). Both files PROGRAMFILE and FILENAME must
        actually exist for this to work, but only PROGRAMFILE
        needs to be a registered program. #vFILENAME_PROTOCOL#} |
         v:vMSG
627 j:lh1_present
628 l:lN_h32
629
630 f!:{^(program)$}:lN_h33
631 v:vCMD | z!: | v:vCMD #use upercase..
632 r@:{#vCMD# <~ Executes #vCMD# as a system command if it
        exists --- first, if it is any of the standard commands
        in TEA OS, otherwise if it is a valid filename "#vCMD#"
        that is already recognized/registered as a user-created
        program - this later case similar to calling "run #vCMD
        #", otherwise shows error message command-not-found.} | v
        :vMSG
633 j:lh1_present
634 l:lN_h33
635
636 # the user-program catch-all...
637 f!:{^([a-z0-9_]+(\.[a-z0-9]+)?)$}:lN_h34
638 v:vPROGRAMFILE
639 y@:dfProgramExists | v:fProgramExists #import
640 y:vPROGRAMFILE | e*:fProgramExists
641 f!:{^PROGRAM EXISTS$}:lN_h34
642 r@:{#vPROGRAMFILE# <~ Is a valid user-registered [TEA]
        program. You might run or execute it by directly invoking
         "#vPROGRAMFILE#" or via "run #vPROGRAMFILE#".} | v:vMSG
643 j:lh1_present
644 l:lN_h34
645
646 l:lh1_present
```

```
y@:dvDELIM2|v:vD
g*:{}:vD:vMSG:vD
"

L:lSYS_HELP_END
#:{=================}

*:{==|FILE SYSTEM |==}
L:lSYS_FILE_START

#--[FILE SYSTEM Functions]

#f(N)->DFS_N
V:fGetFilenameProtocolMSG:"i!:{Valid FILENAME should consist
    of ONLY letters a-zA-Z, numbers 0-9, underscore _ and/or
    a dot .}"
Y:fGetFilenameProtocolMSG|V@:dfGetFilenameProtocolMSG

#f(N)->DFS_N
V:fGenDBFilename:"x:{DFS_}"
Y:fGenDBFilename|V@:dfGenDBFilename

#f(DFS_N)->N
V:fDecodeDBFilename:"d:^DFS_"
Y:fDecodeDBFilename|V@:dfDecodeDBFilename

#f(N)->0|1
V:fIsFileNameValid:"
f!:^[a-z0-9_]+(\.[a-z0-9]+)?$:lBAD_FILE_NAME
i!:1|q!:
l:lBAD_FILE_NAME
i!:0
"
Y:fIsFileNameValid|V@:dfIsFileNameValid


#create FILENAME
V:fCreateFile:"
# first, grab incoming function argument...
v:vFILENAME|t!.:|v:vFNAME

#then import functions..
y@:dfGenDBFilename|v:fGenDBFilename
y@:dfGetLatestOutputBuffer|v:fGetLatestOutputBuffer
y@:dfIsFileNameValid|v:fIsFileNameValid

y:vFNAME | f:^$:lNFINE # filename shouldn't be blank
# file name should conform to accepted patterns
y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
r@:{Invalid File Name [#vFNAME#]!
```

```
695 FILE NOT CREATED.}|q!:
696 l:lGOOD_FILENAME
697
698 # fetch current file name list..
699 y@:dvFileList | v:vFileList
700 # ensure file name doesn't yet exist in records
701 #e.g v:vFileList:{F1|F2|F33|File|File}
702 #v:vFNAME:{f1}
703 # check if file name is unique
704 r@:{(^#vFNAME#\|)|(\|#vFNAME#\|)|(\|#vFNAME#$)}|v:
        vFNAME_REGEX
705 y:vFileList | f*!:vFNAME_REGEX:lFNAME_UNIQUE:
        lFNAME_DUPLICATE
706 l:lFNAME_UNIQUE
707 # add name to file list
708 g*:{|}:vFileList:vFNAME | v:vFileList | v@:dvFileList
709 #also create blank file record in db filesystem
710 y:vFNAME|E*:fGenDBFilename|v:vdFNAME
711
712 v@:dvDATA:{} # we'll write empty file
713 r@:{y@:dvDATA|v@:#vdFNAME#}|E:
714
715 i!:{CREATED NEW BLANK FILE: }|x*!:vFNAME
716 q!:
717 l:lFNAME_DUPLICATE
718 i!:{File Already Exists!
719 NOT DUPLICATING FILE: }|x*!:vFNAME
720 q!:
721 l:lNFINE
722 i!:{Invalid [blank] File Name!
723 FILE NOT CREATED.}
724 "
725
726 Y:fCreateFile|V@:dfCreateFile
727
728
729 #delete FILENAME
730 V:fDeleteFile:"
731 # first, grab incoming function argument...
732 v:vFILENAME|t!.:|v:vFNAME
733
734 #then import functions..
735 y@:dfGenDBFilename|v:fGenDBFilename
736 y@:dfIsFileNameValid|v:fIsFileNameValid
737
738 y:vFNAME | f:^$:lNFINE # filename shouldn't be blank
739 # file name should conform to accepted patterns
740 y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
741 r@:{Invalid File Name [#vFNAME#]!
742 FILE NOT DELETED.}|q!:
```

```
l:lGOOD_FILENAME

# fetch current file name list..
y@:dvFileList | v:vFileList
# remove file name from records

#v:vFileList:{F1|F2|F33|F3|File|File}
#v:vFNAME:{File}
# get filename pattern

# create delete filename patterns + apply them..
r@:{(^#vFNAME#\|)|(\|#vFNAME#$)}|v:vFNAME_REGEX_ENDS
r@:{(\|#vFNAME#\|)}|v:vFNAME_REGEX_MID
r@:{^#vFNAME#$}|v:vFNAME_REGEX_ALL
# use it to del
y:vFileList | d*:vFNAME_REGEX_ENDS | v:vFileList
y:vFileList | d*:vFNAME_REGEX_ALL | v:vFileList
v:vDIV:{|}
r*:vFileList:vFNAME_REGEX_MID:vDIV | v:vFileList

# update file list
y:vFileList | v@:dvFileList
#also delete file record from db filesystem
y:vFNAME|E*:fGenDBFilename|v:vdFNAME
r@:{c!@:#vdFNAME#}|E:

i!:{DELETED FILE: }|x*!:vFNAME
q!:

l:lNFINE
i!:{Invalid [blank] File Name!
FILE NOT DELETED.}
"

Y:fDeleteFile|V@:dfDeleteFile

#ls | list | dir | files
V:fShowFilesList:"
# takes no parameters ;)
# fetch current file name list..
y@:dvFileList
#process and present..
h!:{\|}|d:{\|}
"

#(ls | list | dir | files | find) PATTERN
V:fSearchFilesList:"
# takes file regex pattern parameter
v:vREGEX_PATTERN
# fetch current file name list..
```

```
y@:dvFileList
#process and present..
h!:{\|}|d:{\|}
# only keep what matches...
k*:vREGEX_PATTERN
"

#(del | delete | rm) PATTERN
V:fDeleteFilesList:"
# takes file regex pattern parameter
v:vREGEX_PATTERN

#imports..
y@:dfDeleteFile | v:fDeleteFile
y@:dfProgramExists | v:fProgramExists

# fetch current file name list..
y@:dvFileList
#process and present..
h!:{\|}|d:{\|}
# only keep what matches...
k*:vREGEX_PATTERN
g:{|} | v:vDelFileList | v:vDelFileListORIG
f:^$:NO_FILES_TO_DELETE
# to track files we couldn't delete
v:vNOTDelFileList:{}

# go through files and delete..
v:vNCOUNTER:0
v:vNCOUNTER_DEL:0

l:lSTART_DEL
y:vDelFileList
f:^$:lDONE_DEL
#fetch..
y:vDelFileList | d!:{^[^\|]+\|?}
d:{\|} | v:vDelFileNAME

#if actually blank.. go pop
f:^$:lPOPDEL_LIST

# ensure it is not a registered program
y:vDelFileNAME | e*:fProgramExists | f:{^PROGRAM EXISTS$}:
    lDELETE_FAILED

#r@:{--Continuing to delete file [#vDelFileNAME#]} | q!:

# delete file..
y:vDelFileNAME | e*:fDeleteFile
#q!: # check...
```

```
842
843 f!:{DELETED FILE}:lDELETE_FAILED:lDELETED_OK
844 l:lDELETE_FAILED
845 # add name to NOT deleted file list
846 g*:{|}:vNOTDelFileList:vDelFileNAME  | v:vNOTDelFileList
847 j:lDONE_DELETED_OK
848 l:lDELETED_OK
849 y:vNCOUNTER_DEL | x!:+1 | r.: | v:vNCOUNTER_DEL
850 l:lDONE_DELETED_OK
851 y:vNCOUNTER | x!:+1 | r.: | v:vNCOUNTER
852
853 #pop..
854 l:lPOPDEL_LIST
855 y:vDelFileList | d:{^[^\|]+\|?} | v:vDelFileList
856 #x:{Remaining: } | q!:
857 j:lSTART_DEL # iterate
858
859 l:lDONE_DEL
860 y:vDelFileListORIG|d:{^\|}:{\|$}|v:vDelFileListORIG
861 y:vNOTDelFileList|d:{^\|}:{\|$}|v:vNOTDelFileList
862 r@:{#vNCOUNTER# - #vNCOUNTER_DEL#}|r.:|v:vNCOUNTER_NDEL
863 y:vNCOUNTER_DEL | f:^1$:lDEL_REP_1
864 r@:{DELETED #vNCOUNTER_DEL# file[s] from list:
865 #vDelFileListORIG#
866 EXCEPT #vNCOUNTER_NDEL#:
867 #vNOTDelFileList#}|q!:
868
869 l:lDEL_REP_1
870 y:vDelFileListORIG|d:{^\|}:{\|$}|v:vDelFileListORIG
871 y:vNCOUNTER | f!:^1$:lDEL_REP_2
872 r@:{DELETED ONLY #vNCOUNTER_DEL# file:
873 #vDelFileListORIG#
874 }|q!:
875
876 l:lDEL_REP_2
877 y:vDelFileListORIG|d:{^\|}:{\|$}|v:vDelFileListORIG
878 y:vNOTDelFileList|d:{^\|}:{\|$}|v:vNOTDelFileList
879 r@:{DELETED ONLY #vNCOUNTER_DEL# file from list:
880 #vDelFileListORIG#
881 EXCEPT #vNCOUNTER_NDEL#:
882 #vNOTDelFileList#}|q!:
883
884 l:NO_FILES_TO_DELETE
885 r@:{#vREGEX_PATTERN# matched no FILES!
886 NO FILE WAS DELETED}
887 "
888
889 #init FILENAME
890 V:fInitFile:"
891 v:vFILENAME|t!.:|v:vFNAME
```

```
892
893 #imports...
894 y@:dfCreateFile|v:fCreateFile
895 y@:dfGenDBFilename|v:fGenDBFilename
896 y@:dfGetLatestOutputBuffer|v:fGetLatestOutputBuffer
897 y@:dfIsFileNameValid|v:fIsFileNameValid
898
899 y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
900 r@:{Invalid File Name [#vFNAME#]!
901 FILE NOT CREATED.}|q!:
902 l:lGOOD_FILENAME
903
904 # then, try to create the file...
905 y:vFNAME | E*:fCreateFile
906 # check if it was created...
907 f!:{CREATED NEW BLANK FILE}:lFILE_NOT_CREATED
908 # get name of that newly created file...
909 y:vFNAME|E*:fGenDBFilename|v:vdFNAME
910
911 # get current output buffer
912 y@:dvDATABUFFER | v@:dvDATA
913
914 # only if it was empty then...
915 f!:^$:lFINE_DATA_BUFFER
916 # use print buffer...
917 e*:fGetLatestOutputBuffer | v@:dvDATA
918 l:lFINE_DATA_BUFFER
919
920 # store it into file system record...
921 r@:{y@:dvDATA|v@:#vdFNAME#}|E:
922
923 #a test...
924 #r@:{y@:#vdFNAME#}|E:
925 #x:{File now contains:
926 #}|q!:
927
928 i!:{CREATED and INITIALIZED NEW File: }|x*!:vFNAME
929 q!:
930 l:lFILE_NOT_CREATED
931 i!:{FAILED TO CREATE FILE: } | x*!:vFNAME
932 "
933
934 #write FILENAME
935 V:fWriteFile:"
936 v:vFILENAME|t!.:|v:vFNAME
937
938 #imports...
939 y@:dfCreateFile|v:fCreateFile
940 y@:dfGenDBFilename|v:fGenDBFilename
941 y@:dfGetLatestOutputBuffer|v:fGetLatestOutputBuffer
```

```
y@:dfIsFileNameValid|v:fIsFileNameValid

y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
r@:{Invalid File Name [#vFNAME#]!
FILE NOT WRITTEN.}|q!:
l:lGOOD_FILENAME

# then, try to create the file...
y:vFNAME | E*:fCreateFile
# check if it was created...
f!:{(CREATED NEW BLANK FILE)|(NOT DUPLICATING FILE)}:
    lFILE_NOT_CREATED
# get name of that file...
y:vFNAME|E*:fGenDBFilename|v:vdFNAME
# get current output buffer
y@:dvDATABUFFER | v@:dvDATA

# only if it was empty then...
f!:^$:lFINE_DATA_BUFFER
# use print buffer...
e*:fGetLatestOutputBuffer | v@:dvDATA
l:lFINE_DATA_BUFFER

# store it into file system record...
r@:{y@:dvDATA|v@:#vdFNAME#}|E:

i!:{DATA WRITTEN to File: }|x*!:vFNAME
q!:
l:lFILE_NOT_CREATED
i!:{FAILED TO WRITE FILE: } | x*!:vFNAME
"

#savedata FILENAME
#reads special buffer v@:dvDATABUFFER
V:fSaveDataToFile:"
v:vFILENAME|t!.:|v:vFNAME

#imports...
y@:dfCreateFile|v:fCreateFile
y@:dfGenDBFilename|v:fGenDBFilename
y@:dfGetLatestOutputBuffer|v:fGetLatestOutputBuffer
y@:dfIsFileNameValid|v:fIsFileNameValid

y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
r@:{Invalid File Name [#vFNAME#]!
FILE NOT WRITTEN.}|q!:
l:lGOOD_FILENAME

# then, try to create the file...
y:vFNAME | E*:fCreateFile
```

```
991  # check if it was created...
992  f!:{(CREATED NEW BLANK FILE)|(NOT DUPLICATING FILE)}:
         lFILE_NOT_CREATED
993  # get name of that file...
994  y:vFNAME|E*:fGenDBFilename|v:vdFNAME
995  # get current data buffer
996  # and store it into file system record...
997  r@:{y@:dvDATABUFFER|v@:#vdFNAME#}|E:
998
999  i!:{DATA WRITTEN to File: }|x*!:vFNAME
1000 q!:
1001 l:lFILE_NOT_CREATED
1002 i!:{FAILED TO WRITE FILE: } | x*!:vFNAME
1003 "
1004
1005 #read FILENAME
1006 V:fReadFile:"
1007 v:vFILENAME|t!.:|v:vFNAME
1008
1009 #imports...
1010 y@:dfGenDBFilename|v:fGenDBFilename
1011 y@:dfIsFileNameValid|v:fIsFileNameValid
1012
1013 y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
1014 r@:{Invalid File Name [#vFNAME#]!
1015 FILE NOT READ.}|q!:
1016 l:lGOOD_FILENAME
1017
1018 # then, try to read the file...
1019 y:vFNAME|E*:fGenDBFilename|v:vdFNAME
1020
1021 r@:{y@:#vdFNAME#}|E:
1022 "
1023
1024 y:fReadFile | v@:dfReadFile
1025
1026 #exists FILENAME
1027 V:fFileExists:"
1028 # first, grab incoming function argument...
1029 v:vFILENAME|t!.:|v:vFNAME
1030
1031 #then import functions..
1032 y@:dfGenDBFilename|v:fGenDBFilename
1033 y@:dfIsFileNameValid|v:fIsFileNameValid
1034
1035 y:vFNAME | f:^$:lNFINE
1036 y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
1037 r@:{Invalid File Name [#vFNAME#]!
1038 FILE MAY NOT EXIST.}|q!:
1039 l:lGOOD_FILENAME
```

```
1040
1041 # fetch current file name list..
1042 y@:dvFileList | v:vFileList
1043
1044
1045 #v:vFileList:{F1|F2|F33|F3|File|File}
1046 #v:vFNAME:{f1}
1047
1048 # create filename patterns..
1049 r@:{(^#vFNAME#\|)|(\|#vFNAME#$)}|v:vFNAME_REGEX_ENDS
1050 r@:{(\|#vFNAME#\|)}|v:vFNAME_REGEX_MID
1051 r@:{^#vFNAME#$}|v:vFNAME_REGEX_ALL
1052 # use them to test for existence
1053 y:vFileList | f*:vFNAME_REGEX_ENDS:lFILE_EXISTS
1054 f*:vFNAME_REGEX_ALL:lFILE_EXISTS
1055 f*:vFNAME_REGEX_MID:lFILE_EXISTS
1056
1057 i!:{FILE DOES NOT EXIST}|q!:
1058
1059 l:lFILE_EXISTS
1060 i!:{FILE EXISTS}
1061 q!:
1062
1063 l:lNFINE
1064 i!:{Invalid [blank] File Name!
1065 FILE DOES NOT EXIST.}
1066 "
1067
1068 L:lSYS_FILE_END
1069 #:{=================}
1070
1071
1072 *:{==|PROGRAMMING SYSTEM |==}
1073 L:lSYS_PROG_START
1074
1075 #--[PROGRAMMING SYSTEM Functions]
1076
1077
1078 #runnable FILENAME
1079 V:fRegisterProgramFile:"
1080 # we assume given program file already exists!
1081 # first, grab incoming function argument...
1082 v:vFILENAME|t!.:|v:vFNAME
1083
1084 #then import functions..
1085 y@:dfGenDBFilename|v:fGenDBFilename
1086 y@:dfIsFileNameValid|v:fIsFileNameValid
1087
1088 y:vFNAME | f:^$:lNFINE
1089 y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
```

```
r@:{Invalid File Name [#vFNAME#]!
CANNOT REGISTER PROGRAM.}|q!:
l:lGOOD_FILENAME

# fetch current program file name list..
y@:dvProgramsList | v:vProgramsList

#v:vFileList:{F1|F2|F33|F3|File|File}
#v:vFNAME:{f1}

# check if program name is unique:
# create filename patterns...
r@:{(^#vFNAME#\|)|(\|#vFNAME#\|)|(\|#vFNAME#$)}|v:
    vFNAME_REGEX
# use them to test for program existence
y:vProgramsList | f*!:vFNAME_REGEX:lPROGNAME_UNIQUE:
    lPROG_EXISTS
l:lPROGNAME_UNIQUE
# So, program isn't registered yet, meaning its ok
# to proceed to register it...
# add name to programs list
g*:{|}:vProgramsList:vFNAME | v:vProgramsList | v@:
    dvProgramsList
r@:{User PROGRAM [#vFNAME#] is now REGISTERED.
CAN BE INVOKED as SYSTEM COMMAND.} | q!:

l:lPROG_EXISTS
r@:{User PROGRAM [#vFNAME#] Already EXISTS!
CANNOT re-REGISTER PROGRAM.} | q!:

l:lNFINE
i!:{Invalid [blank] File Name!
CANNOT REGISTER PROGRAM.}
"


#programs
V:fShowProgramsList:"
y@:dvProgramsList|h!:{\|}|d:{\|}
"

#programs PATTERN
V:fSearchProgramsList:"
v:vREGEX_PATTERN
y@:dvProgramsList|h!:{\|}|d:{\|}|k*:vREGEX_PATTERN
"

#run FILENAME
V:fRunFile:"
v:vFILENAME|t!.:|v:vFNAME
```

```
1137
1138 #imports...
1139 y@:dfGenDBFilename|v:fGenDBFilename
1140 y@:dfIsFileNameValid|v:fIsFileNameValid
1141
1142 y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
1143 r@:{Invalid File Name [#vFNAME#]!
1144 FILE NOT EXECUTED.}|q!:
1145 l:lGOOD_FILENAME
1146
1147 # then, try to run the file...
1148 y:vFNAME|E*:fGenDBFilename|v:vdFNAME
1149
1150 # for now, run it as a TEA program..
1151 r@:{y@:#vdFNAME#}|E:|E:
1152 "
1153
1154 #(isrunnable|canrun) FILENAME
1155 V:fProgramExists:"
1156 v:vFILENAME|t!.:|v:vFNAME
1157
1158 y@:dfGenDBFilename|v:fGenDBFilename
1159 y@:dfIsFileNameValid|v:fIsFileNameValid
1160
1161 y:vFNAME | f:^$:lNFINE
1162 y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
1163 r@:{Invalid File Name [#vFNAME#]!
1164 PROGRAM MAY NOT EXIST.}|q!:
1165 l:lGOOD_FILENAME
1166
1167 # fetch current programs name list..
1168 y@:dvProgramsList | v:vProgramsList
1169
1170 r@:{(^#vFNAME#\|)|(\|#vFNAME#\|)|(\|#vFNAME#$)}|v:
        vFNAME_REGEX
1171 y:vProgramsList | f*:vFNAME_REGEX:lPROG_EXISTS
1172
1173 i!:{PROGRAM DOES NOT EXIST}|q!:
1174
1175 l:lPROG_EXISTS
1176 i!:{PROGRAM EXISTS}|q!:
1177
1178 l:lNFINE
1179 i!:{Invalid [blank] File Name!
1180 PROGRAM DOES NOT EXIST.}
1181 "
1182
1183 Y:fProgramExists | V@:dfProgramExists
1184
1185
```

```
#unregister FILENAME
V:fUnRegisterProgram:"
# we assume given program file already exists!
# first, grab incoming function argument...
v:vFILENAME|t!.:|v:vFNAME

#then import functions..
y@:dfGenDBFilename|v:fGenDBFilename
y@:dfIsFileNameValid|v:fIsFileNameValid

y:vFNAME | f:^$:lNFINE
y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
r@:{Invalid File Name [#vFNAME#]!
CANNOT UNREGISTER PROGRAM.}|q!:
l:lGOOD_FILENAME

# fetch current program file name list..
y@:dvProgramsList | v:vProgramsList

# create delete filename patterns + apply them..
r@:{(^#vFNAME#\|)|(\|#vFNAME#$)}|v:vFNAME_REGEX_ENDS
r@:{(\|#vFNAME#\|)}|v:vFNAME_REGEX_MID
r@:{^#vFNAME#$}|v:vFNAME_REGEX_ALL
# use it to del
y:vProgramsList | d*:vFNAME_REGEX_ENDS | v:vProgramsList
y:vProgramsList| d*:vFNAME_REGEX_ALL | v:vProgramsList
v:vDIV:{|}
r*:vProgramsList:vFNAME_REGEX_MID:vDIV | v:vProgramsList

# update programs list
y:vProgramsList| v@:dvProgramsList

i!:{UNREGISTERED PROGRAM FILE: }|x*!:vFNAME
q!:

l:lNFINE
i!:{Invalid [blank] File Name!
PROGRAM FILE NOT UNREGISTERED.}
"

Y:fUnRegisterProgram | V@:dfUnRegisterProgram

#delete-program FILENAME
V:fDeleteProgramFile:"
# first, grab incoming function argument...
v:vFILENAME|t!.:|v:vFNAME

#then import functions..
y@:dfGenDBFilename|v:fGenDBFilename
y@:dfIsFileNameValid|v:fIsFileNameValid
```

```
y@:dfUnRegisterProgram |v:fUnRegisterProgram

y:vFNAME | f:^$:lNFINE # filename shouldn't be blank
# file name should conform to accepted patterns
y:vFNAME | E*:fIsFileNameValid | f!:0:lGOOD_FILENAME
r@:{Invalid File Name [#vFNAME#]!
PROGRAM FILE NOT DELETED.}|q!:
l:lGOOD_FILENAME

# fetch current file name list..
y@:dvFileList | v:vFileList


# create delete filename patterns + apply them..
r@:{(^#vFNAME#\|)|(\|#vFNAME#$)}|v:vFNAME_REGEX_ENDS
r@:{(\|#vFNAME#\|)}|v:vFNAME_REGEX_MID
r@:{^#vFNAME#$}|v:vFNAME_REGEX_ALL
# use it to del
y:vFileList | d*:vFNAME_REGEX_ENDS | v:vFileList
y:vFileList | d*:vFNAME_REGEX_ALL | v:vFileList
v:vDIV:{|}
r*:vFileList:vFNAME_REGEX_MID:vDIV | v:vFileList

# update file list
y:vFileList | v@:dvFileList
#also delete file record from db filesystem
y:vFNAME|e*:fGenDBFilename|v:vdFNAME
r@:{c!@:#vdFNAME#}|E:

# then unregister program file
y:vFNAME | e*:fUnRegisterProgram | v:vUnRegStatus

r@:{DELETED PROGRAM FILE: #vFNAME#
With Status: #vUnRegStatus#} | q!:


l:lNFINE
i!:{Invalid [blank] File Name!
PROGRAM FILE NOT DELETED.}
"

Y:fDeleteProgramFile|V@:dfDeleteProgramFile

L:lSYS_PROG_END
#:{=================}


*:{==|SYSTEM SHELL |==}
L:lSYS_SHELL_START
E*:fLogin
```

```
1286
1287 #---[ Main LOOP ]
1288 L:lSTART_Main
1289
1290 # present default prompt..
1291 E*:fMakePS
1292 V@:dvLastPS
1293 E*:fPROMPT
1294
1295 E*:fLogAI
1296
1297 #clean+process input..
1298 V:vIN_raw
1299 t!.:| v:vIN_clean
1300 Z:|v:vIN_clean_lc
1301
1302 #what did we get?
1303 y:vIN_clean_lc
1304
1305 #preliminary processing...
1306
1307 # a subsystem..
1308 ###((FILE System))
1309
1310 #> create FILENAME
1311 F!:^create\s+.*$:lN_CRT_FILE
1312 Y:vIN_raw|t!.:
1313 V@:dvLastCommand
1314 d:^\S+\s+|V:vFileName
1315 E*:fCreateFile | E*:fLog0
1316 J:lFIN_CMD_PROCESSING
1317 L:lN_CRT_FILE
1318
1319 #> ls | list| dir | files
1320 F!:{^(ls|list|dir|files)$}:lN_LIST_FILES
1321 V@:dvLastCommand
1322 E*:fShowFilesList | E*:fLog0
1323 J:lFIN_CMD_PROCESSING
1324 L:lN_LIST_FILES
1325
1326 #> (ls | list | dir | files | find) PATTERN
1327 F!:{^(ls|list|dir|files|find)\s+(\S+\s*)+$}:lN_SEARCH_FILES
1328 Y:vIN_raw|t!.:
1329 V@:dvLastCommand
1330 d:^\S+\s+|V:vFilePattern
1331 E*:fSearchFilesList | E*:fLog0
1332 J:lFIN_CMD_PROCESSING
1333 L:lN_SEARCH_FILES
1334
1335 #> (del | delete | rm) PATTERN
```

```
F!:{^(del|delete|rm)\s+(\S+\s*)+$}:lN_DEL_FILES
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFilePattern
E*:fDeleteFilesList | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_DEL_FILES

#> init FILENAME
F!:^init\s+.*$:lN_INIT_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileName
E*:fInitFile | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_INIT_FILE

#> write FILENAME
F!:^write\s+.*$:lN_WRITE_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileName
E*:fWriteFile | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_WRITE_FILE

#> (show|cat|read) FILENAME
F!:{^(show|cat|read)\s+.*$}:lN_READ_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileName
E*:fReadFile | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_READ_FILE


#> exists FILENAME
F!:{^(exists)\s+.*$}:lN_EXISTS_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileName
E*:fFileExists | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_EXISTS_FILE


#> (rename|rn|mv|move) FILENAME NEWNAME
F!:{^(rename|rn|mv|move)\s+([a-z0-9_]+(\.[a-z0-9]+)?)\s+([a-
    z0-9_]+(\.[a-z0-9]+)?)$}:lN_RENAME_FILE
Y:vIN_raw|t!.:
```

```
V@:dvLastCommand
d:^\S+\s+|V:vFileNames
Y:vFileNames | D:\s+.*$ | V:vFileName
Y:vFileNames | D!:\s+.*$ | V:vNewName
#check..
#R@:{Renaming #vFileName# to #vNewName#} | q!:
# ensure source file exists...
Y:vFileName | E*:fFileExists
F!:{^FILE EXISTS$}:lCANNOT_RENAME

# also check if it is a registered program...
Y:vFileName | E*:fProgramExists
F:{^PROGRAM EXISTS$}:lPROCESS_RENAME_PROGRAM

# renaming normal file...
# read src into data buffer
Y:vFileName | E*:fReadFile | v@:dvDATABUFFER
# write data buffer into new file
Y:vNewName | E*:fSaveDataToFile | v:vWRITELOG
# delete old file
Y:vFileName | E*:fDeleteFile
R@:{File #vFileName# RENAMED #vNewName#
With status: #vWRITELOG#}
E*:fLog0
J:lFIN_CMD_PROCESSING

L:lPROCESS_RENAME_PROGRAM
# renaming program file...
# read src into data buffer
Y:vFileName | E*:fReadFile | v@:dvDATABUFFER
# write data buffer into new file
Y:vNewName | E*:fSaveDataToFile | v:vWRITELOG
# register new program
Y:vNewName | E*:fRegisterProgramFile | v:vREGLOG
# delete old program file
Y:vFileName | E*:fDeleteProgramFile | v:vDELLOG
R@:{Program File #vFileName# RENAMED #vNewName#
With [WRITE] status: #vWRITELOG#
And [REGISTER] status: #vREGLOG#
And [DEL] status: #vDELLOG#}
E*:fLog0
J:lFIN_CMD_PROCESSING

L:lCANNOT_RENAME
R@:{File #vFileName# does NOT EXIST!
RENAME Operation ABANDONED}
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_RENAME_FILE

```

```
#> (copy|clone) FILE1 FILE2
F!:{^(copy|clone)\s+([a-z0-9_]+(\.[a-z0-9]+)?)\s+([a-z0-9_
    ]+(\.[a-z0-9]+)?)$}:lN_CLONE_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileNames
Y:vFileNames | D:\s+.*$ | V:vFileName
Y:vFileNames | D!:\s+.*$ | V:vFileName2
#check..
#R@:{Cloning #vFileName# to #vFileName2#} | q!:
# ensure source file exists...
Y:vFileName | E*:fFileExists
F!:{^FILE EXISTS$}:lCANNOT_CLONE
# read src into data buffer
Y:vFileName | E*:fReadFile | v@:dvDATABUFFER
# write data buffer into new file
Y:vFileName2 | E*:fSaveDataToFile | v:vWRITELOG
R@:{File #vFileName# CLONED TO #vFileName2#
With status: #vWRITELOG#}
E*:fLog0
J:lFIN_CMD_PROCESSING

l:lCANNOT_CLONE
R@:{File #vFileName# does NOT EXIST!
CLONE Operation ABANDONED}
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_CLONE_FILE


#> export FILENAME
F!:^export\s+.*$:lN_EXPORT_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
#exit via file export..
d:^\S+\s+|V:vFileName
E*:fReadFile | v:vLOG # read file
E*:fResetLog # reset log
# Gracefully quit...
E*:fOSHandle | X:{Thanks for Using }
V:vMSG
G*!:vDELIM2:vFileName:vLOG:vMSG
J:lEXIT_Main
L:lN_EXPORT_FILE

###((END FILE System))

# a subsystem..
###((INTERFACE System))

```

```
#> uimini
F!:^uimini$:lN_UIMINI
V@:dvLastCommand
V@:dvUIMODE_MINI:{1}
I!:{TOS Mini-Interface Mode Activated} | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_UIMINI

#> uilog
F!:^uilog$:lN_UILOG
V@:dvLastCommand
V@:dvUIMODE_MINI:{0}
I!:{TOS Log-Interface Mode Activated} | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_UILOG

###((END INTERFACE System))


# a subsystem..
###((HELP System))

#> about
F!:^about$:lN_ABOUT
V@:dvLastCommand
E*:fGetAboutTOS | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_ABOUT

#> help | man | whatis | info
F!:{^(help|man|whatis|info)$}:lN_HELP
V@:dvLastCommand
E*:fGetHelpMenu | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_HELP

#> (help|man|whatis|info|about) NAME
F!:{^(help|man|whatis|info|about)\s+(\S+\s*)+$}:
    lN_HELP_TOPIC
Y:vIN_raw|t!.:
V@:dvLastCommand
#extract+process topic..
d:{^\S+\s+} | E*:fGetHelpOn | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_HELP_TOPIC

#> help-file[s]?
F!:^help-file[s]?$:lN_HELP_FILE
|V@:dvLastCommand
E*:fGetFileCMDMenu | E*:fLog0
```

```
1533 J:lFIN_CMD_PROCESSING
1534 L:lN_HELP_FILE
1535
1536 #> help-prog[s|rams]?
1537 F!:{^help-prog([s]|rams)?$}:lN_HELP_PROG
1538 V@:dvLastCommand
1539 E*:fGetProgrammingMenu | E*:fLogO
1540 J:lFIN_CMD_PROCESSING
1541 L:lN_HELP_PROG
1542
1543 #> help-util[s]?
1544 F!:^help-util[s]?$:lN_HELP_UT
1545 V@:dvLastCommand
1546 E*:fGetUtilityMenu | E*:fLogO
1547 J:lFIN_CMD_PROCESSING
1548 L:lN_HELP_UT
1549
1550 #> help-ui[s]?
1551 F!:^help-ui[s]?$:lN_HELP_UI
1552 V@:dvLastCommand
1553 E*:fGetUIMenu | E*:fLogO
1554 J:lFIN_CMD_PROCESSING
1555 L:lN_HELP_UI
1556
1557 #> help-help[s]?
1558 F!:^help-help[s]?$:lN_HELP_HELP
1559 V@:dvLastCommand
1560 E*:fGetHelpMenu
1561 X:{
1562 To use or access help on any topic in the system, use any of
      the commands presented in the following Menu.
1563 }
1564 X!:{
1565 Each of those commands leads you to further information
     about help on other commands. For example
1566
1567 > help help-util
1568
1569 Will explain what the help-util command is all about.}
1570
1571 E*:fLogO
1572 J:lFIN_CMD_PROCESSING
1573 L:lN_HELP_HELP
1574
1575 ###((END HELP System))
1576
1577 # a subsystem..
1578 ###((UTILITY System))
1579
1580 #> echo MESSAGE
```

```
F!:^echo.*$:lN_ECHO
Y:vIN_raw|t!.:
d:^\S+\s*|V:vMESSAGE
X:{echo }
|V@:dvLastCommand
Y:vMESSAGE
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_ECHO

#> (fetch|web) URL
F!:{^(fetch|web)\s+\S*$}:lN_FETCH
Y:vIN_raw|t!.:
V@:dvLastCommand
#extract url..
d:{^\S+\s+}
E*:fFixURL
W: #do the web..
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_FETCH

#> tea CODE
F!:^tea\s+[\S]+.*$:lN_TEA
Y:vIN_raw|t!.:
#extract code..
d:^\S+\s+|V:vTEACode
X:{tea }
|V@:dvLastCommand
Y:vTEACode
E: #do the tea..
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_TEA

#> math EXPRESSION
F!:^math\s+[\S]+.*$:lN_MATH
Y:vIN_raw|t!.:
#extract expression..
d:^\S+\s+|V:vMathExp
X:{math }
|V@:dvLastCommand
Y:vMathExp
R.: #do the math..
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_MATH

#> username NAME
F!:^username\s+[\S]+.*$:lN_SETUSER
```

```
V@:dvLastCommand
#extract expression..
Y:vIN_raw|t!.:
d:^\S+\s+
V@:dvUSERNAME #update db
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_SETUSER

#> (whoami|name)
F!:{^(whoami|name)$}:lN_WHOAMI
V@:dvLastCommand
Y@:dvUSERNAME
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_WHOAMI


#> time
F!:^time$:lN_TIME
v@:dvLastCommand
#show cur.time..
Z.:TIME | E*:fLog0
#present+log I/O
J:lFIN_CMD_PROCESSING
L:lN_TIME

#> date
F!:^date$:lN_DATE
v@:dvLastCommand
Z.:DATE | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_DATE

#> timestamp
F!:^timestamp$:lN_TSTAMP
v@:dvLastCommand
Z.:TIMESTAMP | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_TSTAMP

#> clear | reset
F!:{^(clear|reset)$}:lN_CLEAR
v@:dvLastCommand
E*:fResetLog
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_CLEAR

#> print
```

```
F!:^print$:lN_PRINT
V@:dvLastCommand
E*:fGetLatestOutputBuffer
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_PRINT

#> logout
F!:^logout$:lN_LOGOUT
C!@:dvUSERNAME # delete username
C*:vUSERNAME # also clear vault
E*:fResetLog # reset log
# ensure to clear log backup too!
C@:dvLastOutput_backup
# Gracefully quit...
E*:fOSHandle | X:{You have SIGNED OUT!
Thanks for Using }
J:lEXIT_Main
L:lN_LOGOUT

#> exit
F!:^exit$:lN_EXIT
#don't log input cmd.
E*:fResetLog # reset log
#prepare for immediate exit..
# Gracefully quit...
E*:fOSHandle | X:{Thanks for Using }
J:lEXIT_Main
L:lN_EXIT

#> export
F!:^export$:lN_EXPORT
#don't log input cmd.
#exit via export..
Y@:dvLOG|V:vLOG
E*:fResetLog # reset log
# Gracefully quit...
E*:fOSHandle | X:{Thanks for Using }
V:vMSG
G*!:vDELIM2:vLOG:vMSG
J:lEXIT_Main
L:lN_EXPORT

###((END UTILITY System))


# a subsystem..
###((PROGRAMMING System))

#> (runnable|register|add-program) FILENAME
```

```
F!:{^(runnable|register|add-program)\s+([a-z0-9_]+(\.[a-z0
    -9]+)?)$}:lN_RUNNABLE_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileName
#check..
#R@:{Registering User-Program in #vFileName#} | q!:
# ensure source file exists...
Y:vFileName | E*:fFileExists
F!:{^FILE EXISTS$}:lCANNOT_REGISTER_PROG
# register file as user-program..
Y:vFileName | E*:fRegisterProgramFile
E*:fLog0
J:lFIN_CMD_PROCESSING

L:lCANNOT_REGISTER_PROG
R@:{File #vFileName# does NOT EXIST!
User-Program Registration Operation ABANDONED}
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_RUNNABLE_FILE


#> programs
F!:{^programs$}:lN_LIST_PROGRAMS
V@:dvLastCommand
E*:fShowProgramsList | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_LIST_PROGRAMS

#>programs PATTERN
F!:{^programs\s+(\S+\s*)+$}:lN_SEARCH_PROGRAMS
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFilePattern
E*:fSearchProgramsList | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_SEARCH_PROGRAMS

#> (run|exec|execute) FILENAME
F!:{^(run|exec|execute)\s+.*$}:lN_RUN_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileName
E*:fRunFile | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_RUN_FILE

#> (isrunnable|canrun) FILENAME
F!:{^(isrunnable|canrun)\s+.*$}:lN_CANRUN_FILE
```

```
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileName
E*:fProgramExists | E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_CANRUN_FILE


#> (del-prog|delete-program|unregister) FILENAME
F!:{^(del-prog|delete-program|unregister)\s+([a-z0-9_]+(\.[a
    -z0-9]+)?)$}:lN_UNREGISTER_FILE
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileName
#check..
#R@:{Unregistering User-Program in #vFileName#} | q!:
# ensure source file exists...
Y:vFileName | E*:fFileExists
F!:{^FILE EXISTS$}:lCANNOT_REGISTER_PROG
# unregister user-program..
Y:vFileName | E*:fUnRegisterProgram
E*:fLog0
J:lFIN_CMD_PROCESSING

L:lCANNOT_REGISTER_PROG
R@:{File #vFileName# does NOT EXIST!
User-Program Unregistration Operation ABANDONED}
E*:fLog0
J:lFIN_CMD_PROCESSING
L:lN_UNREGISTER_FILE


#> (update|update-program) FILE1 FILE2
F!:{^(update|update-program)\s+([a-z0-9_]+(\.[a-z0-9]+)?)\s
    +([a-z0-9_]+(\.[a-z0-9]+)?)$}:lN_UPDATE_PROGRAM
Y:vIN_raw|t!.:
V@:dvLastCommand
d:^\S+\s+|V:vFileNames
Y:vFileNames | D:\s+.*$ | V:vFileName
Y:vFileNames | D!:\s+.*$ | V:vUpdateFileName
#check..
#R@:{Updating #vFileName# using #vUpdateFileName#} | q!:

# ensure source file exists...
Y:vUpdateFileName | E*:fFileExists
F!:{^FILE EXISTS$}:lCANNOT_READ_SRC

# ensure target file exists...
Y:vFileName | E*:fFileExists
F!:{^FILE EXISTS$}:lCANNOT_WRITE_TARGET
```

```
1828
1829 # also ensure it is a registered program...
1830 Y:vFileName | E*:fProgramExists
1831 F:{^PROGRAM EXISTS$}:lPROCESS_UPDATE_PROGRAM
1832
1833 R@:{Target File #vFileName# IS NOT a PROGRAM!
1834 Program UPDATE Operation ABANDONED}
1835 E*:fLog0
1836 J:lFIN_CMD_PROCESSING
1837
1838 L:lPROCESS_UPDATE_PROGRAM
1839 # updating program file...
1840 # read src into data buffer
1841 Y:vUpdateFileName | E*:fReadFile | v@:dvDATABUFFER
1842 # write data buffer into existing program file
1843 Y:vFileName  | E*:fSaveDataToFile | v:vWRITELOG
1844
1845 R@:{Program File #vFileName# UPDATED USING #vUpdateFileName#
1846 With [WRITE] status: #vWRITELOG#}
1847 E*:fLog0
1848 J:lFIN_CMD_PROCESSING
1849
1850 L:lCANNOT_WRITE_TARGET
1851 R@:{Target File #vFileName# does NOT EXIST!
1852 Program UPDATE Operation ABANDONED}
1853 E*:fLog0
1854 J:lFIN_CMD_PROCESSING
1855
1856 L:lCANNOT_READ_SRC
1857 R@:{Source File #vUpdateFileName# does NOT EXIST!
1858 Program UPDATE Operation ABANDONED}
1859 E*:fLog0
1860 J:lFIN_CMD_PROCESSING
1861 L:lN_UPDATE_PROGRAM
1862
1863 #> PROGRAM | FILENAME
1864 F!:{^([a-z0-9_]+(\.[a-z0-9]+)?)\s*$}:lN_RUN_FILE_PROG
1865 Y:vIN_raw|t!.:
1866 V@:dvLastCommand | V:vFileName
1867
1868 # ensure it is a registered program, otherwise cnf..
1869 Y:vFileName | E*:fProgramExists
1870 F:{^PROGRAM EXISTS$}:lPROCESS_RUN_PROGRAM:lCMD_CATCHALL
1871
1872 L:lPROCESS_RUN_PROGRAM
1873 Y:vFileName | E*:fRunFile | E*:fLog0
1874 J:lFIN_CMD_PROCESSING
1875 L:lN_RUN_FILE_PROG
1876
1877 ###((END PROGRAMMING System))
```

```
1878
1879
1880 ###((CATCH-ALL))
1881 L:lCMD_CATCHALL
1882 #if we came this far
1883 #then it's CNF!
1884 F:^$:lFIN_CMD_PROCESSING
1885 Y:vIN_raw | V@:dvLastCommand # log unknown command
1886 I!:{--COMMAND NOT FOUND | INVALID COMMAND--}
1887 E*:fLog0
1888 J:lFIN_CMD_PROCESSING
1889 ###((END CATCH-ALL))
1890
1891 #we only naturally get here for properly
1892 #processed commands..
1893 L:lFIN_CMD_PROCESSING
1894
1895 #for non-exit commands, we want to ensure
1896 #we log user response from latest command..
1897 E*:fLogAI
1898
1899 J:lSTART_Main
1900 ####[End Main LOOP]
1901
1902 #outside shell.. quitting
1903 L:lEXIT_Main
1904
1905 #delete session log from db
1906 C!@:dvLOG
1907
1908 Q!: #quit+return AI
1909
1910 L:lSYS_SHELL_END
1911 #:{=================}
```

Listing 1: TEA Program: TOS v1.1.0 Complete Source-Code

6 About TEA OS v1.1.0 and TEA v1.5.4

Anyone interested in trying out not just the most stable TEA language release, but also its hottest application/use-case to date needs get their hands on the TEA OS without doubt. A preview of what has been accomplished so far can readily be accessed via the TEA WEB IDE: `https://tea.nuchwezi.com`, however, the specifics concerning that particular milestone are:

The latest official reference implementation of the TEA runtime and documentation has reached v1.5.4. That version of TEA has been robustly tested and proven, being the one currently powering the final, minimal, feature-complete,

self-documenting TEA Operating System (TEA OS) also currently being included as part of the official TTTT package; also at v1.5.4 right now. Thus, not only can one use TEA for the 'usual' tasks (refer to the TEA TAZ [1]), but with TEA OS, one has a complete TEA-powered file-system, a suite of basic utilities including ability to do math and test simple and complex minified TEA programs from the TOS shell terminal, and the power to write, store, and later invoke stored TEA programs as system commands directly from TEA OS.

Also, among great observations while working with TEA OS is that all work done and stored can not only be readily exported outside of the OS (per session), but also that; files create and stored inside of TEA OS can later be updated, renamed, deleted or exported outside of the TEA OS as is. Concerning this, it should be worth noting that, unlike most other operating systems (big and small), that often require some kind of virtual machine in order to be run from inside a host operating system, that the TEA OS is actually creatively designed (thanks to TEA), so that, anywhere TEA programs can run, the TEA OS can likewise be run — without need for any modern or traditional virtualization technologies in-between the TEA OS and say Linux or Windows! However, it should also be worth noting that, despite this, files created or read from inside the TEA OS aren't immediately or readily visible to the host operating system; that is, when inside TEA OS, you can not run a command like "ls" and expect to see a listing of files hosted on the underlying operating system. Similarly, you can not be inside Windows or Linux, and expect to readily see, modify or read files created inside of the TEA OS! This is should then indicate that TEA and TEA OS somewhat allow for a means of virtualization that somewhat goes beyond or is acutely different from what normal operating systems implementation might be accustomed to[2].

Like all TEA programs, the complete TEA OS can fit inside a string, and one can carry it as a lightweight, easier-to-use alternative to Linux, Unix or Windows/MS-DOS in just any modern browser or via the TEA mobile IDE without installing anything, no virtual machines, no sophisticated configurations... it just works!

TO try TEA OS without installing anything, use the hot-load-and-run URL:
`https://tea.nuchwezi.com/?fc=https://gist.githubusercontent.com/mcnemesis/276b40c96b52f846bf11214419912e3f/raw/&i=Please+Click+Run+In+Case+It+Does+Not+Auto+Run&run`

The latest documentation on both TEA and TEA OS can be found in [1], and as for what it feels like working inside of the TEA OS right now, the following illustrative session dump might be worth reviewing or referring to..

[2]Thus the joke some TEAists are currently championing, that the "TEA OS is a #beyond-linux OS".

7 TEA OS Shell: an illustrative session dump

TEA OS v1.1.0 Work Session

```
ooooooo_________________
___oo_____ooooo___ooooo__
___oo____oo____o_oo___oo_
___oo____ooooooo_oo___oo_
___oo____oo______oo___oo_
___oo_____ooooo___oooo_o_
_________________________
TEA OS v1.1.0 is Ready...

Please set a Username:
Fut. Prof. JWL
--------------------

Hello, Fut. Prof. JWL. Welcome to TOS!
--------------------
TOS/21:35:42/>
ls
.
datefile
info.text
info
nugreetings
run
the_greetings
permutate
helloworld
about.tea
--------------------
TOS/21:35:49/>
help

=o==o==o==o==o==o==o=
TEA OS v1.1.0 HELP System Commands:
about
(help | man | whatis | info | about) [NAME]
help-file
help-help
help-prog
help-ui
help-util
```

```
=o==o==o==o==o==o==o=


------------------
TOS/21:36:02/>
about

=o==o==o==o==o==o==o=
ABOUT: TEA OS v1.1.0
#---[ABOUT TEA OS]

NAME: TEA Operating System
(also "TEA OS" or just "TOS")

ARCHITECT: J. Willrich Lutalo <jwl@nuchwezi.com>

SINCE: 9th APRIL, 2026

INTENT: To manifest a true general purpose, lightweight,
    minimal but powerful operating system that can
    essentially just fit inside a single string; to see
    what the limits of the mathematics of TRANSFORMATICS
    might be, and what we surely could accomplish with its
    proper use via modern portable computers.

ABOUT: TEA OS is implemented using the Transforming
    Executable Alphabet (TEA) programming language. It is
    based on a specification first laid out in the TEA TAZ.
     It is a project originating from UGANDA.

PROJECT HOME: Nuchwezi Research | https://tea.nuchwezi.com
=o==o==o==o==o==o==o=


------------------
TOS/21:36:15/>
help-file

=o==o==o==o==o==o==o=
TEA OS v1.1.0 FILE System Commands:
create FILENAME
init FILENAME
write FILENAME
(rename | rn | mv | move) FILENAME NEWNAME
(copy|clone) FILE1 FILE2
exists FILENAME
```

```
find PATTERN
(show | cat | read) FILENAME
export FILENAME
(del | delete | rm) FILENAME
(ls | list | dir | files) PATTERN
=o==o==o==o==o==o==o=


--------------------
TOS/21:36:25/>
help-help

To use or access help on any topic in the system, use any
    of the commands presented in the following Menu.

=o==o==o==o==o==o==o=
TEA OS v1.1.0 HELP System Commands:
about
(help | man | whatis | info | about) [NAME]
help-file
help-help
help-prog
help-ui
help-util
=o==o==o==o==o==o==o=

Each of those commands leads you to further information
    about help on other commands. For example

> help help-util

Will explain what the help-util command is all about.
--------------------
TOS/21:36:37/>
uilog
TOS Log-Interface Mode Activated
--------------------
TOS/21:36:56/>
help uimini

=o==o==o==o==o==o==o=
uimini <~ Configure TEA OS interface to render using mini-
    terminal mode. Setting persists across logins until
    changed. This mode is suitable for environments such as
     using TOS on some Desktop Computer Systems.
```

```
=o==o==o==o==o==o==o=

------------------
TOS/21:37:05/>
uimini
TOS Mini-Interface Mode Activated
------------------
TOS/21:37:11/>
help-prog

=o==o==o==o==o==o==o=
TEA OS v1.1.0 PROGRAMMING System Commands:
(runnable | register | add-program) FILENAME
(isrunnable | canrun) FILENAME
(run | exec | execute) FILENAME
(del-prog | delete-program | unregister) FILENAME
(update | update-program) FILENAME
PROGRAM
(help | man | whatis | info | about) PROGRAM
=o==o==o==o==o==o==o=

------------------
TOS/21:37:18/>
whatis program

=o==o==o==o==o==o==o=
PROGRAM <~ Executes PROGRAM as a system command if it
    exists --- first, if it is any of the standard commands
     in TEA OS, otherwise if it is a valid filename "
    PROGRAM" that is already recognized/registered as a
    user-created program - this later case similar to
    calling "run PROGRAM", otherwise shows error message
    command-not-found.
=o==o==o==o==o==o==o=

------------------
TOS/21:37:34/>
okay
--COMMAND NOT FOUND | INVALID COMMAND--
------------------
TOS/21:37:42/>
help-utils

=o==o==o==o==o==o==o=
```

```
TEA OS v1.1.0 UTILITY System Commands:
clear | reset
date
echo MESSAGE
exit
export
(fetch | web) [URL] [FILENAME]
logout
math EXPRESSION
print
tea CODE
time
timestamp
username NAME
whoami | name
=o==o==o==o==o==o==o=


--------------------
TOS/21:37:48/>
date
Monday, 18 May 2026
--------------------
TOS/21:37:52/>
whoami
Fut. Prof. JWL
--------------------
TOS/21:37:57/>
programs
nugreetings
the_greetings
about.tea
--------------------
TOS/21:38:01/>
cat about.tea
i!:Hello World
--------------------
TOS/21:38:16/>
run about.tea
Hello World
--------------------
TOS/21:38:23/>
tea n!:100|v:vN|r@:{Here's a random number: #vN#}
Here's a random number: 94
--------------------
```

```
TOS/21:39:55/>
echo n!:100|v:vN|r@:{Here's a random number: #vN#}
n!:100|v:vN|r@:{Here's a random number: #vN#}
--------------------
TOS/21:40:11/>
write rng.tea
DATA WRITTEN to File: rng.tea
--------------------
TOS/21:40:30/>
whatis rng.tea

=o==o==o==o==o==o==o=
That help topic [rng.tea] is not understood or supported.
    Try the 'help' command to see what help commands there
    are. For example 'help whatis'
=o==o==o==o==o==o==o=


--------------------
TOS/21:40:38/>
programs
nugreetings
the_greetings
about.tea
--------------------
TOS/21:40:42/>
files
.
datefile
info.text
info
nugreetings
run
the_greetings
permutate
helloworld
about.tea
rng.tea
--------------------
TOS/21:40:48/>
add-program rng.tea
User PROGRAM [rng.tea] is now REGISTERED.
CAN BE INVOKED as SYSTEM COMMAND.
--------------------
TOS/21:41:01/>
```

```
whatis rng.tea

=o==o==o==o==o==o==o=
rng.tea <~ Is a valid user-registered [TEA] program. You
    might run or execute it by directly invoking "rng.tea"
    or via "run rng.tea".
=o==o==o==o==o==o==o=


--------------------
TOS/21:41:07/>
run rng.tea
Here's a random number: 23
--------------------
TOS/21:41:16/>
about move

=o==o==o==o==o==o==o=
move FILENAME NEWNAME <~ Change the name of an existing
    file from FILENAME to NEWNAME. WILL ALSO update the
    registered [user] programs list if FILENAME was a
    registered program. Valid FILENAME should consist of
    ONLY letters a-zA-Z, numbers 0-9, underscore _ and/or a
     dot .
=o==o==o==o==o==o==o=


--------------------
TOS/21:41:29/>
move rng.tea rng
Program File rng.tea RENAMED rng
With [WRITE] status: DATA WRITTEN to File: rng
And [REGISTER] status: User PROGRAM [rng] is now REGISTERED
    .
CAN BE INVOKED as SYSTEM COMMAND.
And [DEL] status: DELETED PROGRAM FILE: rng.tea
With Status: UNREGISTERED PROGRAM FILE: rng.tea
--------------------
TOS/21:41:41/>
whatis rng

=o==o==o==o==o==o==o=
rng <~ Is a valid user-registered [TEA] program. You might
    run or execute it by directly invoking "rng" or via "
    run rng".
=o==o==o==o==o==o==o=
```

```
------------------
TOS/21:41:47/>
rang
--COMMAND NOT FOUND | INVALID COMMAND--
------------------
TOS/21:41:50/>
rng
Here's a random number: 60
------------------
TOS/21:41:53/>
rng
Here's a random number: 76
------------------
TOS/21:41:58/>
time
21:42:04
------------------
TOS/21:42:04/>
name
Fut. Prof. JWL
------------------
TOS/21:42:07/>
export
=o==o==o==o==o==o==o=
Thanks for Using TEA OS v1.1.0
```

8 Conclusion

The Transforming Executable Alphabet has made possible the implementation of a minimal, lightweight, portable and feature-complete software operating environment known as the TEA OS or TOS. TEA OS demonstrates that a compact, high-level runtime and service stack can provide many of the conveniences of an operating system while remaining extremely portable and runnable in environments where traditional kernels cannot [readily] operate. We characterize TEA OS as a minimal OS runtime rather than a conventional kernel; extending it to full kernel status would require adding low-level primitives (boot, drivers, memory protection, scheduling) or a small privileged shim, a direction we identify as future work in case the need should arise, but otherwise which, based on the design and original intentions of TOS, we might not have to worry about.

We have covered highlights of TOS v1.1.0 in this paper, including looking at the underlying system architecture, the complete verbatim source-code of “the operating system that can fit inside of a single string”, as well as looked at a sam-

ple session dump exported from TOSH. The future is very bright! Given TOS can be run without the need for any traditional virtualization technologies such as VirtualBox, Hypervisor, ISOs, etc. It means that one can indeed leverage TEA OS to not only learn how to work inside a standard minimal (terminal-UI operating system), but also that, work typically heavy to carry around even for so-called lightweight operating systems such as Minix, Linux-on-A-Stick/USB, etc, cannot readily rival the ease with which one can create, update, run, post, pull, and export files as well as programs using the TEA operating system from the most unusual of places — inside a web browser, on a typical mobile phone/smartphone! Further work concerning informing the world (both in academia, but also the industry) about what has been accomplished with TEA, but also the TEA OS is much needed, however, and without doubt, the future, or what some used to think of as "the future", is indeed, finally, already here!

References

[1] Joseph Willrich Lutalo. **TEA TAZ - Transforming Executable Alphabet A: to Z: COMMAND SPACE SPECIFICATION**. *Academia.edu*, 2024. `https://doi.org/10.5281/zenodo.20398135`.

[2] Joseph Willrich Lutalo. **TEA: Transforming Executable Alphabet - A Minimal Language Leveraging Sequence Transformer Chaining**. Preprints, EngrXiv, 2026. `https://doi.org/10.31224/7185`.

[3] Joseph Willrich Lutalo, Odongo Steven Eyobu, and Benjamin Kanagwa. **DNAP: Dynamic Nuchwezi Architecture Platform-A New Software Extension and Construction Technology**. *authorea*, 2023. `https://doi.org/10.36227/techrxiv.13176365.v1`.

[4] Ellen Siever, Stephen Spainhour, Stephen Figgins, and Jessica P. Hekman. *Linux in a Nutshell: A Desktop Quick Reference*. O'Reilly Media, Sebastopol, CA, 3rd edition, 2000. Available online at `https://learning.oreilly.com/library/view/linux-in-a-nutshell/0596000251/`.

[5] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, Madison, WI, 2018. Free PDF available at the official site: `https://pages.cs.wisc.edu/~remzi/OSTEP/`.

[6] Richard Y. Kain. *Advanced Computer Architecture: A Systems Design Approach*. Prentice Hall, Englewood Cliffs, NJ, 1999. Digital preview available at Internet Archive: `https://archive.org/details/advancedcomputer0000kain`.

[7] Microsoft Copilot. Research assistance and various clarifications. AI-generated insights via Copilot discussion, 2026. Personal communication, during drafting and revisions of manuscript.