TempleOS
3 May 2023
Emilee Oquist

Background

Created by Terry A. Davis, TempleOS has garnered a cult-like following over the past few years. One resource, TinkerOS - is also a base where TempleOS fans can experiment with Terry's OS on their own computer. It has been exceptionally helpful in finding information on TempleOS's design structure. It differs from TempleOS only in that it made minor changes so that it is more accessible to play with on modern machinery, as some of Terry's work relies on hardware that has been deprecated.

The key to appreciating all of TempleOS's quirks comes with the understanding that all of its design choices had been implement purposefully, and was not just the results of a naive programmer. TempleOS was created with a specific vision in mind that bases itself off of the Commodore 64. Terry's goal for his OS was for it to be purely user developer without the use of 3rd party libraries, built with an upper limit of 100k lines of code, as well as only having 20k read-only memory ROM. Terry's description of TempleOS in his documentation is as follows:

> *"TempleOS is a x86_64, multi-cored, non-preemptive multi-tasking, ring-0-only, single-address_mapped (identity-mapped), operating system for recreational programming. Paging is almost not used."*[1] It is also a non-networked operating system, as stated on the main documentation page.[2]

---

[1] "Welcome to Templeos." The Temple Operating System. Accessed May 1, 2023. https://tinkeros.github.io/WbTempleOS/Doc/Welcome.html#l1.
[2] The temple operating system. Accessed May 1, 2023. https://tinkeros.github.io/WbTempleOS/.

About Terry

Terry A. Davis' work on TempleOS is incredibly impressive, but it should be known that much of his inspiration came after a revelation from God in 1996. Starting in the early 2000s and worked on for more than 14 years, TempleOS had been created to be the Third Temple. Although not originally named TempleOS, biblical references are a common theme in both naming conventions and other structures built into TempleOS. One very prominent case being the programming language specific to TempleOS called HolyC.

Terry was diagnosed as suffering from schizophrenia, which, while giving him inspiration to create such a unique operating system, also led him to having bursts of very socially inappropriate behavior. He was known for making comments deemed as racist, as well as going off on conspiratorial tangents.[3]

Process Management

Unlike commonly used operating systems such as Windows, Linux, or macOS, TempleOS has no thread support because it doesn't use threads at all. This correlates to how TempleOS manages its memory; there is no memory protection whatsoever, which means if you want to parallelize something, then you can share data with another forked process.

In popular operating systems, you will not find these kinds of design choices. For TempleOS, having all processes share a single virtual address space means that context switching is fast and overall lightweight. The reason modern systems don't share address space, and prefer to have

[3] "Terry A. Davis." Wikipedia. Wikimedia Foundation, May 3, 2023. https://en.wikipedia.org/wiki/Terry_A._Davis.

larger overhead costs associated with separate address spaces is to prevent critical failures and crashes. One process crashing or terminating can have a ripple effect across the whole system and cause it to crash as well. A single misused pointer could crash the entire system.[4]

Scheduling Policy

Although not using threads, TempleOS does make use of multi-tasking. Its multitasking is cooperative, meaning that a task must explicitly call Yield to give up the CPU to another task. As per Terry's descriptions, this saves the current process' registers and loads in the next task. This forces users to write their programs in such a way that it cooperates with other processes and doesn't dominate the CPU. This allows programmers to have much more control over how their created programs work, but it can add another level of complexity- making sure not to forget to Yield, or finding the best place to Yield. On another note, running tasks in TempleOS are organized in a doubly-linked list with *ready*, *run*, *blocked*, and *waiting*.[5]

Memory Management

As mentioned previously, TempleOS has no memory protection. All code is ring-0-only. There are no permission levels - ring 0 means that everything runs at the kernel level. In the F.A.Q. section of TempleOs' documentation, Terry does a fantastic job of explaining how memory is managed.

> "*TempleOS identity-maps all memory, all the time. It is like paging is not used. There is no special kernel high half memory space. TempleOS is ring-0-only, so everything is*

[4] "Codersnotes." A Constructive Look At TempleOS. Accessed May 1, 2023. http://www.codersnotes.com/notes/a-constructive-look-at-templeos/.
[5] "Templeos Programs in Linux User-Space, Part 2: Anatomy of a Kernel." minexew's blog, March 29, 2020. https://minexew.github.io/2020/03/29/templeos-loader-part2.html.

kernel, even user programs. There is a special task called Adam and he doesn't die, so his heap never gets freed. That's as close to kernel memory as it gets. All code goes in the lowest 2Gig of addresses, known as the Code Heap, so that the REL32 addressing mode can be used."[6]

On top of not having memory protection, TempleOS also lacks memory segmentation. Memory addresses are represented by 64-bit numbers and TempleOS' memory is flat. This allows the CPU to linearly and directly address all memory locations without handling any typical memory segmentation or paging schemes.

When Terry describes his reasoning for all of his design choices, they tend to all lead back to being told he should do it that way by God. Logistically, however, these design choices make the tradeoff between stability and efficiency. TempleOS is able to be used near-instantaneously because it can boot off the hard disk in about one second.[7]

CONCLUSION

TempleOS is not meant to be the main operating system on a computer, hence why it is lacking in some areas when it comes to functionality, but is rather meant to be fun to tinker with. Coming from a background of loving the Commodore 64, perhaps considering the simplicity of TempleOS as a good thing could be beneficial to those learning how operating systems work. It may have a high learning curve but very little is obfuscated or abstracted away, and having direct

---

[6] "Frequently Asked Questions." The Temple Operating System. Accessed May 3, 2023. https://tinkeros.github.io/WbTempleOS/Doc/FAQ.html#l1.
Briefly referenced throughout the paper. I would personally recommend reading through the F.A.Q. to better understand Terry A. Davis and his vision.

[7] "Codersnotes."

access to everything within TempleOS allows for a more intimate understanding of how certain systems work.

BIBLIOGRAPHY

"Codersnotes." A Constructive Look At TempleOS. Accessed May 1, 2023. http://www.codersnotes.com/notes/a-constructive-look-at-templeos/.

"Frequently Asked Questions." The Temple Operating System. Accessed May 3, 2023. https://tinkeros.github.io/WbTempleOS/Doc/FAQ.html#l1.

The temple operating system. Accessed May 1, 2023. https://tinkeros.github.io/WbTempleOS/.

"Templeos Programs in Linux User-Space, Part 2: Anatomy of a Kernel." minexew's blog, March 29, 2020. https://minexew.github.io/2020/03/29/templeos-loader-part2.html.

"Terry A. Davis." Wikipedia. Wikimedia Foundation, May 3, 2023. https://en.wikipedia.org/wiki/Terry_A._Davis.

"Welcome to Templeos." The Temple Operating System. Accessed May 1, 2023. https://tinkeros.github.io/WbTempleOS/Doc/Welcome.html#l1.