TempleOS: Final Report

Robert Coffey

December 11, 2020

Contents

1 Preface

This document contains the final report for my MUN COMP-2004 research project. It is a more formal compilation of the research document contained in the `research/` directory.

Much of this report was paraphrased or copied over from the research document. More thorough references can be found there, should information about TempleOS from the creator himself be desired.

The information in the research document was obtained from the personal website of Terry Davis, the creator of TempleOS. Links to referenced webpages will be provided.

Some interesting facts such as Terry calling TempleOS "God's Third's Temple" were not stated explicitly on his website, but instead within the vast collection of videos he released discussing programming, operating systems, and his life.

This document was generated using Emacs Org-Mode, and is ideally viewed in the `plaintext` format; although `pdf` and `html` formats have also been included.

2 Introduction to TempleOS

TempleOS was built for programmers, it was written with the goal of keeping the line count down, to make it easy to tinker with. [1] The following quote was obtained from the TempleOS documentation Welcome page, written by the creator of TempleOS: Terry Davis. This is his summary of TempleOS in a single sentence.

> TempleOS is a `x86_64`, multi-cored, non-preemptive multi-tasking, ring-0-only, single-address-mapped (identity-mapped), operating system for recreational programming. – Terry Davis [1]

For a conventional general purpose OS, failure is not an option. This is not a problem for TempleOS, as it is intended to be used alongside Windows or Linux. Failure is an option: if something isn't available in TempleOS, use your other OS instead. [1] This allowed Terry to cherry-pick what features to add, with the intent of keeping TempleOS maximally beautiful.

This report will explore various aspects of TempleOS, such as CPU scheduling, memory management, and the user interface. Additionally, it will contain some discussion regarding the evolution and future of TempleOS, it's place in the operating system landscape, as well as the creator Terry Davis himself.

3 Kernel Research

3.1 System Architecture

TempleOS is intended to run on modern x86 64-bit processors. It is generally dual-booted alongside a more powerful OS, or run inside a virtual machine. Although it can be used independently, TempleOS has no networking. [2] This leads to projects in TempleOS having a workflow involving the usage of programs in both TempleOS and another OS.

TempleOS was written completely from scratch, even the bootloader and compiler. [2] This is not a C compiler, it is a compiler for Terry's own language, a dialect of C named HolyC. HolyC utilizes some helpful Pascal syntax to make a language with the power of C, that can be input and compiled directly in the command line, behaving like an interpreted language such as UNIX bash. [2] TempleOS uses HolyC for the majority of its kernel code, but also as its shell scripting language, relieving programmers from having to memorize an additional language that provides little use outside of the shell. [1]

All processes in TempleOS are known as ring-0-only, meaning there is no user mode nor special memory space for the kernel. TempleOS never switches privilege levels, nor changes address maps. [1] This means all processes in TempleOS are essentially always in kernel mode. This reduces the complexity of the OS and eliminates the overhead of switching modes and address maps. This simplicity gives TempleOS the ability to switch tasks at far greater speeds than conventional operating systems. [3]

Having only kernel mode poses great danger, as one misdirected pointer could crash the system. But this is not much of a concern for TempleOS, as it is intended for recreational programming. In fact the resulting simplicity of allowing the programmer to operate without protection is largely the purpose of TempleOS. [1]

3.2 CPU Scheduling

TempleOS is a multi-cored, non-preemptive multi-tasking operating system. [1] It does master-slave asymmetric multiprocessing. [4] `Core0` is the master, the master core's processes explicitly assign tasks to other cores. The TempleOS scheduler does not move tasks between cores, nor does it preempt any tasks. [4] Processes in TempleOS have no priority. [1] This is very similar to the First-Come-First-Serve scheduling algorithm, with the master core behaving as the scheduler.

In TempleOS, each core has a Seth Task that is immortal and is the father of all tasks on that core. [4] The Adam Task, or Adam, refers to the Seth Task on the master core. Adam is the father of all the other Seth Tasks, and begins executing at start-up. [1] Adam is responsible for scheduling and dispatching tasks to the other Seth Tasks. Every Seth Task including Adam has a queue of processes with the Seth Task as the head, each executed as a non-preemptive round-robin queue. [5]

> Each core has an executive Seth Task which is the father of all tasks on that core. Adam is the Seth Task on `Core0`. – Terry Davis [4]

> In TempleOS, Adam Task refers to the father of all tasks. He's never supposed to die. – Terry Davis [1]

> This is Adam, as in Adam and Eve, the parent of all tasks. – Terry Davis [6]

In TempleOS there is only one address map per core, making context switches orders of magnitude faster than conventional operating systems. [3] The TempleOS kernel takes advantage of this by utilizing extra threads for helping to render windows; and through a process called `Spawn`, the kernel can dispatch tasks to other cores and retrieve their results with ease. [4]

3.3 Memory Management

TempleOS tasks inherit the symbols of their parent, thus everything that must be system-wide is associated with Adam, the father of all tasks. [1] Since Adam is immortal, on Adam's heap go all memory objects which are not to be destroyed by any single task's death. [6] Adam's heap is the equivalent of kernel memory in a conventional operating system.

In TempleOS there is no distinction between a process, task, or thread. Each task has a code and data heap which is returned to its parent Seth Task when it dies. [6] Any core can allocate from any heap in any task at any time, even making independent heaps – memory heaps not owned by any task. [6] TempleOS has no concept of kernel and user memory, all memory can be accessed by any process.

> TempleOS identity-maps all memory, all the time. It is like paging is not used. There is no special kernel high half memory space. – Terry Davis [2]

> TempleOS is ring-0-only, so everything is 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. – Terry Davis [2]

TempleOS imposes no protection over memory, which can be dangerous as much like with kernel mode, one misdirected pointer could crash the system. But this is of little concern as TempleOS is intended for recreational programming. [1]

3.4 User Interface

As a personal choice made by Terry Davis, TempleOS exclusively displays in the 640x480 resolution with 16-bit color. [1] TempleOS has a screen refresh rate of 30000/1001 frames-per-second. This is how often screen memory is updated, and it is not synchronized with hardware. [2]

TempleOS has its own sprite format which supports 2D and 3D sprites, 3D sprites are stored as a mesh of triangles. [2] All text files in TempleOS can store sprites directly in the text file. This is done by storing binary sprite data beyond the terminating NULL in the file. [2] These sprites can be seen directly inside the text when editing a file, similar to a `pdf`.

Adam is created at start-up and appears in a small window always available beneath the user's windows. [6] There can be only one window per task, and only tasks on the master core can have windows. [6] Although other cores may help the master core render them. [4]

4 Discussion

4.1 Evolution of TempleOS

TempleOS is just the most recent in a line of hand-rolled operating systems written by Terry Davis. Around 1993, Terry got a 486 processor and was eager to try 32-bit mode; so he wrote a DOS program in TASM that changed to protected mode and never returned to DOS. [7]

This program was called "Terry's Protected Mode OS", TPMOS. This name was inspired by the machines he worked on at the time, which used the "VAXTMOS" operating system. For all intents and purposes, TPMOS is the root of TempleOS' ancestry. TPMOS never got much further than 0xB8000 text mode, echoing the keyboard to screen, simple multitasking, and what was barely a `malloc`. [7] But it was the beginning of what would become Terry's greatest work.

TPMOS was set aside for some time, until 2003 when it was resurrected. Terry used FreeDOS and Visual Studo to compile and execute it, and continued building the OS from there. Around this time he had started a company "H.A.R.E.", and renamed TPMOS to HOPPY. [7]

Next came the challenge of building a proper command line. Terry wanted it to use the same scripting language as he would be creating for the OS. Fueled by hatred for bash scripting, Terry created HolyC: an amalgamation of some helpful Pascal syntax, with the power of C and C++. [2] Giving users a language that can be used for both controlling the command line, as well as writing programs.

> The only problem was, I hated Unix Bash scripting. I could never remember it. As a regular C/C++ programmer, you don't really use bash often enough to memorize it. I thought, "What if I just use C/C++ for scripting!" – Terry Davis [7]

Terry suffered from issues related to mental health throughout his life, but sometime after 2003, his mental health began to decline drastically. He suffered from hallucinations of God, paranoia about the CIA, and became obsessed with his operating system. His hallucinations guided his work. He took upon a mission from what he thought was God: **To create a divine operating system.**

> In 2003, God told me to stick to 640x480 16 color. – Terry Davis [7]

> I didn't start the operating system as a work for God, but He directed my path along the way and kept saying it was His temple. – Terry Davis [7]

His operating system took on multiple names: Doors, Davos, J, LoseThos, SparrowOS, and finally, TempleOS. Along the way he wrote his own bootloaders, compiler, and every other program in TempleOS, abandoning DOS entirely. [7]

> Still I hesistated and kept it secular until, finally, Microsoft went nuclear with SecureBoot and UEFI. Then, I went nuclear and named it "TempleOS". I will command them on orders from God to UNDO THAT STUFF! – Terry Davis [7]

That leaves us with the TempleOS of today, known by Terry Davis as God's Third Temple.

4.2 Future of TempleOS

Terry was the sole programmer of TempleOS, having written the entire thing from scratch; all the way down to the bootloader. [2] Unfortunately, Terry ended his own life in 2018. Without his vision and genius, TempleOS lacks direction.

His project lives on through people like myself, who take interest in his work and who he was, but there is little hope for seeing it continue to develop. TempleOS was and is entirely Terry, this was his idea of how a computer should function.

Without Terry, changing TempleOS would be like vandalism. Using TempleOS feels like you're experiencing a personal space, seeing the old ideas of someone that can only be called a Grand Wizard; someone who endured the woes of uncontrollable intellect that a Grand Wizard would be expected to.

4.3 TempleOS in the OS Landscape

TempleOS was built with programmers, and the Commodore 64, in mind. Having been inspired by how accessible it was to program the C64: Terry designed his platform to minimize the obstacles between the programmer and their compiler.

Although there is little commercial use for TempleOS, it provides a unique platform for tinkerers to have free reign over their machine. Something that just isn't possible in an OS which is to be used by businesses and for critical operations. TempleOS will never run on a hospital network, but it will give you total freedom as a programmer.

Through a combination of the scripting environment provided by the TempleOS command line, and HolyC: programmers can build useful programs with the performance of a compiled language, and the workflow creature comforts of an interpreted language such as UNIX bash.

5 Conclusion

TempleOS was built for programmers, and it shows in every design decision Terry made. TempleOS makes no effort to protect the programmer from themself, and will not complain when you point the gun at your own foot; but it also gives you total freedom to write programs in the way you see fit.

TempleOS' style of inputting code into the terminal and having it sent directly to the compiler allows you to write programs alongside usual shell operation, both using the HolyC language. This provides a unqiue view on the direction programming could have taken, should operating systems like UNIX not have been so successful among programmers and businesses alike.

It's a shame TempleOS was never brought to completion, but what was left behind is an epitome to the skills of its creator, Terry Davis. He wrote the entire operating system from scratch, bootloader onwards; and for programmers like myself, the unique approach he took with every function of TempleOS cracked the seal on what was possible.

6 References

1. Terry Davis. *Welcome to TempleOS.* https://templeos.holyc.xyz/Wb/Doc/Welcome.html
2. Terry Davis. *Frequently Asked Questions.* https://templeos.holyc.xyz/Wb/Doc/FAQ.html
3. Terry Davis. *TempleOS' Features.* https://templeos.holyc.xyz/Wb/Doc/Features.html
4. Terry Davis. *Multi-Core.* https://templeos.holyc.xyz/Wb/Doc/MultiCore.html
5. Terry Davis. *Scheduler.* https://templeos.holyc.xyz/Wb/Kernel/Sched.html
6. Terry Davis. *Glossary.* https://templeos.holyc.xyz/Wb/Doc/Glossary.html
7. Terry Davis. *TempleOS History.* https://templeos.holyc.xyz/Wb/Home/Web/History.html