TempleOS:
A Technical Report

By:
Jesse Arstein (n97k254), Alexis Tingey (n11n497),
Benjamin Barnett (v85f475), Kevin Kleiman (h72r171)

Important Statement From the Team:

It is important to note that due to Davis' schizophrenia, he has made extremely racist, homophobic, and antisemetic comments. We, as a team do not condone this at all. We do not believe that anything, even schizophrenia, is an excuse for racism and bigotry. We felt this was important to point out before moving forward with this report.

Introduction

TempleOS is an operating system designed and developed by Terry A. Davis. The operating system is a 64 bit operating system that was meant to be the Third Temple prophesied within the Bible. The entirety of the OS follows major religious themes and is written in a custom language known as HolyC, also developed by Terry A. Davis.

The purpose of this technical report is to look at a few of the technical aspects of TempleOS and understand the design choices that went into creating an operating system.

History

Terry A. Davis

To understand the history of TempleOS, one must first understand its creator, Terry Davis. Davis was a brilliant, yet troubled man. Davis was born and raised in West Allis, WI, the son of two loving parents. Davis was interested in computers from a young age, learning to write Assembly at a young age for the Commodore64. The Commodore 64 quickly became Davis' favorite computing machine and greatly inspired the design of TempleOS. Davis went on to attend Arizona State University where he studied electrical engineering. After graduating in 1994, Davis then went to work as a programmer at Ticketmaster where he stood out as a brilliant engineer. This would not last long, however, because in 1996 Davis experienced his first manic episode. He began to believe government organizations were after him and that God was speaking to him directly. Most of his manic episodes centered around Davis believing that he was God's messenger and architect, which directly influenced the idea and design around TempleOS. Davis spent most of his time in and out of psychiatric wards until 2003, when he was officially diagnosed with schizophrenia. This illness would only get worse in the years to come and eventually become the reason he developed TempleOS. Davis' delusions devolved into bitter racism, anti semitism, and general bigotry. This was due to his deteriorating mental health, however is not an excuse for this behavior. Some believe that these were not Davis' true thoughts and that his brilliance overshadows his deplorable words, however no amount of brilliance can ever overshadow words like that. Davis was periodically homeless and/or incarcerated in the latter parts of his life. He would frequently wander from his sister's or parents' house in Arizona

TempleOS

Davis originally began development on what was then called J Operating System in order to recreate parts and functions found on most Commodore 64 software. However, after receiving what he perceived to be another direct message from God, Davis renamed the project TempleOS due to his belief that it was to be God's Third Temple on Earth. Religion became the basis of almost every piece of TempleOS, from the nomenclature, to the core functions of libraries and software. There are many biblical references in TempleOS, some hidden, some very obvious and it is clear that Davis' schizophrenia influenced much of this project. If one is able to see past a lot of the religious functions and connotations that TempleOS has, they would be able to see a complex and brilliantly designed operating system that both Linux, Windows, and MacOS could learn something from. TempleOS has some incredibly intelligent design concepts the likes of which most modern operating systems will never have, not to mention it's ultra-lightweight! TempleOS was not designed to be safe or intuitive for the everyday user, however, for the user who can navigate and understand its religious undertones and complex systems, there is an immense amount of power to be found. Through the mess that was it's development and design , one will find the tragic genius of TempleOS.

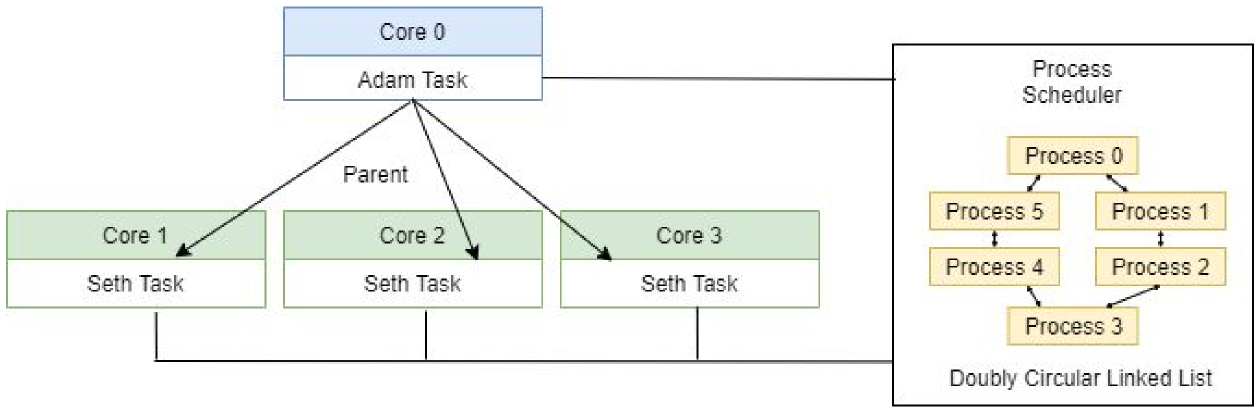
Technical Analysis

Processes and Process Scheduling

The concept of threads doesn't exist within TempleOS. When you spawn a new thread you're actually just generating another process that has access to all the same resources. The operating system is built up from the main process that Terry calls the "Adam Task", in reference to Adam from the biblical tale of Adam and Eve. This process is the parent of all processes within the operating system. The Adam task is immortal until the operating system exits. Items that the user wants to keep in memory that don't disappear when any single task ends should be added to Adam's heap.

One of TempleOS's features is multi core support. Multi core processing is handled in a master-slave design pattern. This is done by assigning a process to each core that handles process scheduling on each core. Terry calls these processes a "Seth Task" in reference to Adam and Eve's son Seth. If you have 4 cores you would have four Seth task's that handle process scheduling on each core with Adam being the Seth Task on core 0.

The operating system uses a circular doubly linked list with a Seth task as the head of the list for each core. Job scheduling is then processed in a round robin fashion. The job scheduler goes to each task and sees if it is in a ready state or a non ready state. If the task is ready the job scheduler will run the task and move onto the next task. Switching contexts of the processes is a matter of updating the registers with the processes's data instead of finding the data in page tables. Terry states due to this design choice tasks can fully switch tasks in half a microsecond. The job scheduler will also check if a task is waiting for a certain period of time and see if it is ready to run. This appears to help not leave any tasks waiting too long for CPU time. While tasks are running on the processor core they will continue to run until a call to the Yield() function is made. Once the Yield() function is called the scheduler will then move onto the next task in the list.



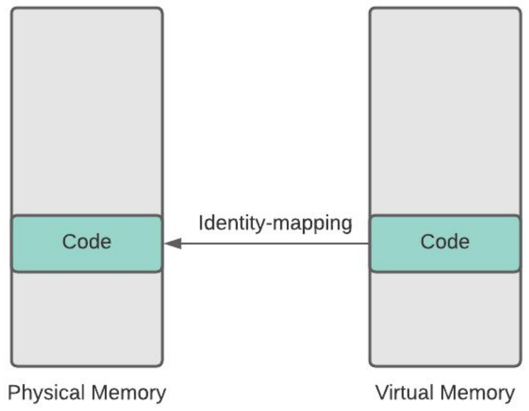
Terry's implementation of locks is meant only for "read-modify-write" statements and if used incorrectly can crash the entire operating system if a lock isn't released correctly or there is a deadlock situation.

TempleOS tracks both parent and children processes. This is done to prevent any zombie child processes from continuing to run even though its parent process has exited. When the kill command is given to a parent process or if a parent process finishes running, TempleOS will handle the ending of any child processes that were spawned off of that parent.

TempleOS exclusively utilizes ring 0 when running programs (Mitton). Since there are no protections built into the operating system every program essentially has access to the same shared resources. Each process has their own contexts that keep track of their own stacks and heaps but the operating system doesn't prevent access to a pointer that doesn't belong to a process. You can randomly manipulate data anywhere you want from your program. Terry designed this operating system to be used by one singular computer with no connectivity to any other computers. He believed the power should belong to the user to do whatever they wanted to do within the operating system.

Memory Management

Davis' idea that his OS be controlled largely by a single user, is apparent in the way memory is handled within TempleOS. He wanted to build a very fast system based on simplicity and complete user access. Terry aimed to avoid paging completely, but unfortunately, 64-bit mode requires paging. He eventually implemented a work around, identity-mapping (shown below) all memory, virtual identical to physical. This leads to the existence of is a singular page table map shared across all tasks. Another aspect of this system's simplicity is that nothing is swapped to disk. Hardware support is nonexistent for TempleOS, other than the core PC system.



The simplicity of identity-mapping led to some issues and ended up being revised in 2016. Davis turned away from his original memory approach and implemented double mapping of the lowest memory within TempleOS, using an uncached alias. This allows memory to be write-through and uncached for different devices. As the OS stands today, the first 4Gig of memory can be accessed without caching.

Again putting control of TempleOS in its user's hands, there does not exist a kernel memory space. In fact, there is no kernel in the way most operating systems have a kernel. Everything, including user programs, is considered "kernel". As was previously mentioned, the closest thing to a kernel within TempleOS is the Adam Task. This task never dies, meaning its heap is never freed, which is the closest thing the system has to kernel memory.

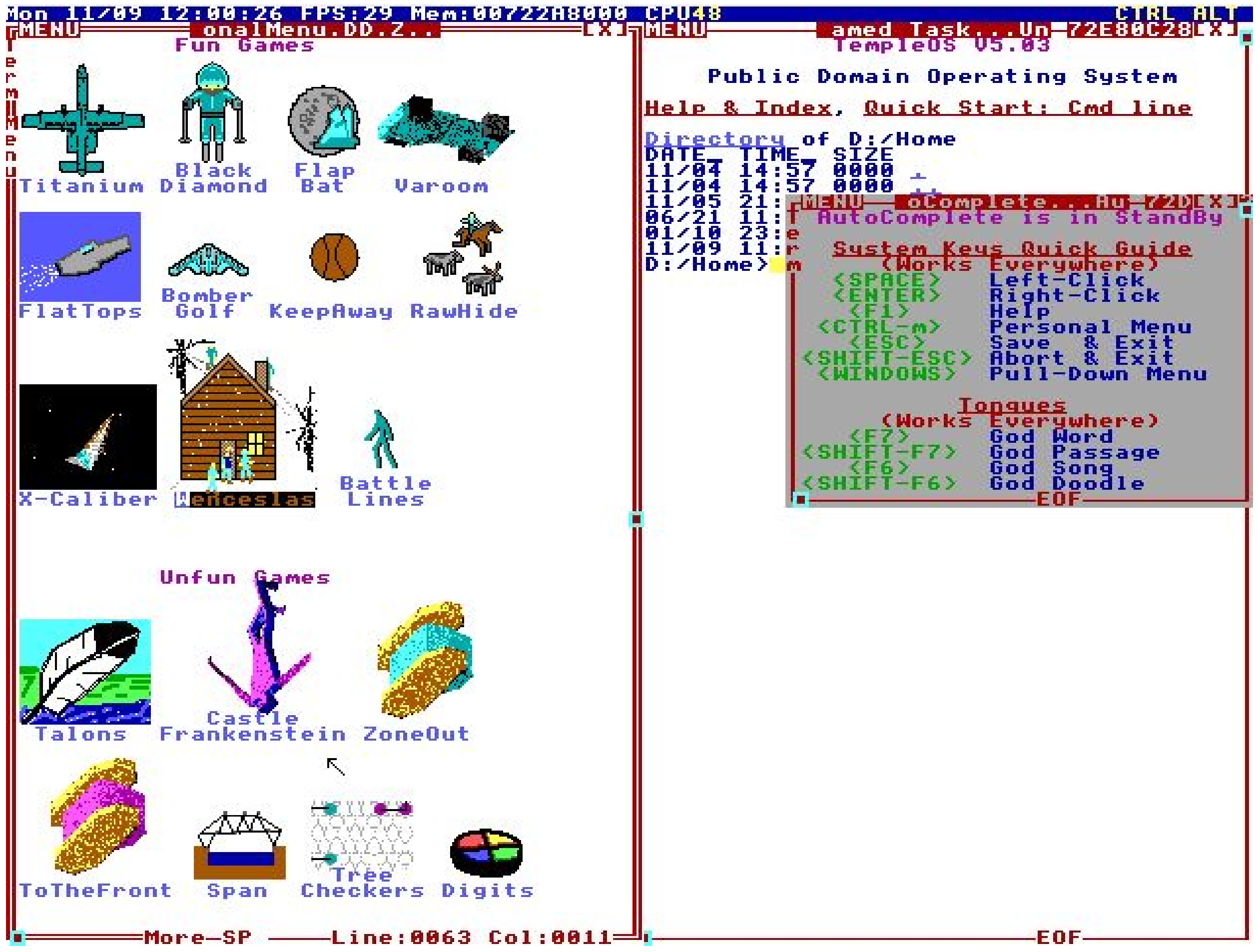
TempleOS allocates memory, which is not segmented, as needed. For example, memory is allocated as more items are displayed in the user interface window. Memory for code is allocated as code is compiled at the command line. Many users believe the system leaks memory, which is possible considering the user is responsible for both freeing memory and killing tasks. Pieces of memory stick to tasks even when their memory is freed, until the running task is killed. The freed memory cannot be used by the task again unless the same size chunk is needed. This leads to memory space hanging in the balance between being freed, and being reused by the same task. To be used by another task, it must wait until its process or task is killed. This design was intentional, leaving the power (or downfalls) of TempleOS largely in the hands of its user.

Usage

Since TempleOS is such a small OS and does not allow for networking, its uses are limited. The creator, Terry Davis, once described Linux as a semi truck, Windows as a car, and Temple as a motorcycle. This pretty much deems it as a recreational OS. On the Welcome page of the official site, Davis states that "The people who can most benefit are" professionals doing hobby projects, teenagers doing projects, and non-professional, older-persons projects”. This reinforces the fact that it's strictly for recreational purposes even more.

Built in Features

I decided to download Temple OS for myself, and was quite surprised with the amount of preloaded programs and games. I counted 28 programs in total.



```
TempleOS V5.03

   Public Domain Operating System

Help & Index, Quick Start: Cmd line

Directory of D:/Home
DATE  TIME  SIZE
11/04 14:57 0000
11/04 14:57 0000
11/05 21:
06/21 11:
01/10 23:
11/09 11:
D:/Home>

AutoComplete is in StandBy

    System Keys Quick Guide
      (Works Everywhere)
   <SPACE>     Left-Click
   <ENTER>     Right-Click
    <F1>       Help
  <CTRL-m>     Personal Menu
    <ESC>      Save  & Exit
 <SHIFT-ESC>   Abort & Exit
  <WINDOWS>    Pull-Down Menu

           Tongues
      (Works Everywhere)
    <F7>       God Word
 <SHIFT-F7>    God Passage
    <F6>       God Song
 <SHIFT-F6>    God Doodle
```

If you look through the Welcome page on the official site, Terry Davis talks alot about simplicity and experimentation. It seems his main goal of the system was to make a fun version of an old operating system. He states on the same page “It's a kayak, not a Titanic -- it will crash if you do something wrong. You quickly reboot, however. DOS and the 8-bit home computers of the 80's worked fine without memory protection and most computers in the world -- the embedded ones -- operate without protection. The resulting simplicity of no protections is why TempleOS has value. In fact, that's the point of TempleOS.”. It seems as if Davis had a nostalgic soft-spot for OS’s that werent stupid proof. If you go to the “Temple OS Charter” on his site, he talks about Solomon's temple and how it was a space to meditate and cherish by beautifying it with more art. You could probably guess that further down the page he states that Temple OS is god's temple. His belief in this is evident from the 18 fully developed games that he blessed the temple with. He also gives links to the course code for each app and says they more or less form a basis that spans the range of use that TempleOS is intended for

Holy C and the Command Line

Temple OS has a pretty unique Holy C editor. It's made such that the command line feeds right into the compiler. To edit a file, one just types Ed("filename"); into the command line. Davis stated this was to keep things simple. He also made it such that anything defined within a global context is immediately executed, kind of like Python. For type casting in Holy C, he uses different, yet logical, prefixes than C or C++.

U0	void, but ZERO size!
I8	char
U8	unsigned char
I16	short
U16	unsigned short
I32	int
U32	unsigned int
I64	long (64-bit)
U64	unsigned long (64-bit)
F64	double

In addition to this, there is no type checking when the code is compiled. This goes along with Davis's mantra of simplicity. In another attempt for simplicity, it allows printing by just putting a char string in parentheses on a line.
"Hello World!\n"; - prints Hello World
He also made a new list of operator precedences, which he supposedly got from the bible.
In short, Davis reinvented C to be better fit for simple single machine, single thread usage. He also included some of the lords, along with his own preferences in its design.

Discussion

Some of the lessons we learned as a group is that when designing an operating system there are a multitude of different design choices you can make. Each of these decisions affect the efficiency, security and usability of the operating system. Terry designed his operating system with the idea that giving power to the user was the most important aspect of the operating system. Due to this overall goal, TempleOS is an insecure operating system. Security is entirely dependent on the user. Other operating systems will have different overarching goals. The big three operating systems each are designed with different goals in mind. Linux prides itself on giving the user more power while also providing a usable system that is secure. MacOS is a different flavor of Linux that abstracts away some of the power a user has in favor of more usability. Windows also is designed with average user usability in mind but also tries to make its platform as available as possible for developers. Backwards compatibility is a large component of Windows operating systems and is part of the reason why it has the largest market share in the average computer user market.

Each of these big operating systems have different philosophies that cater to different subsets of users. One of the interesting things about diving into the design choices of TempleOS is it helped us understand some of the reasoning why each operating system might value a different approach to the management of computer hardware. These operating systems are always developing against the newest hardware and trying to take advantage of any feature a new piece of hardware has. Operating system development is an ongoing process and will continue to try to achieve their design goals in the most efficient way possible.

Conclusion

TempleOS is an impressive testament to the dedication one person can have. An operating system is a complex system that is tasked with managing many different components. It needs to do this efficiently and securely as well. The slightest design mistake can introduce performance problems, security issues and stifle usability. Davis took on this project and managed to create a complex system. There are so many parts of his operating system that we could have dove into. If we continued to analyze TempleOS we would need to look at hundreds of files and thousands of lines of code. There are many aspects of this operating system that we encourage you to look deeper into.

Bibliography

Davis, Terry A. “TempleOS Documentation.” *tempos.holyc.xyz*, Terry A. Davis, 30 March 2018, https://templeos.holyc.xyz/Wb/Doc/. Accessed 1 November 2020.

Mitton, Richard. “A Constructive Look At TempleOS.” *Coders Notes*, Coders Notes, 8 June 2015, http://www.codersnotes.com/notes/a-constructive-look-at-templeos/. Accessed 1 November 2020.