Universidad Nacional Autónoma de México

Facultad de Ingeniería

Division de Ingenieria Electrica

Sistemas operativos

Profesor: Ing. Gunnar Eyal Wolf Iszaevich

Grupo: 06

Exposición

TempleOS: Las implicaciones de un SO operando en el anillo 0 de protección.

Integrantes:

Talonia Fuentes Jesús

Juárez Valdivia Barvara Caridad

Semestre 2025-2

TempleOS: Las implicaciones de un SO operando en el anillo 0 de protección

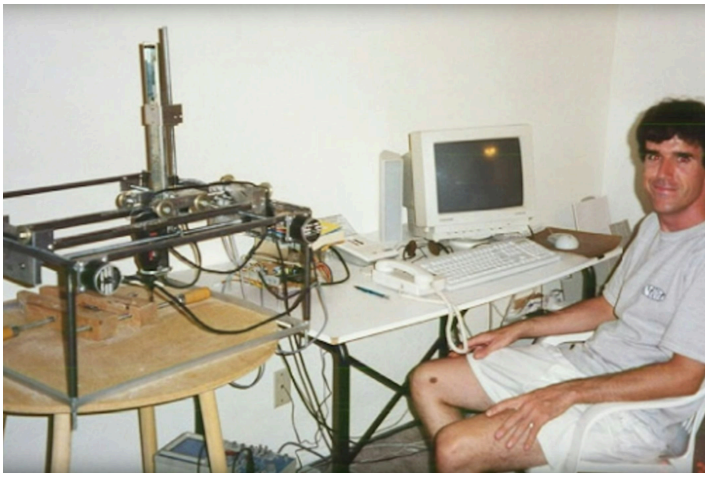
TempleOS (anteriormente conocido como J Operating System, LoseThos y SparrowOS) es un sistema operativo desarrollado por Terry A. Davis (1969-2018) a principios de los años 2000. Aunque no es (y nunca ha sido) muy popular, es reconocido por sus excéntricas y peculiares características derivadas del contexto de su creador.

Se trata de un SO monolítico y monousuario de 64 bits, descrito como 'una modernización de Commodore64', escrito completamente en lenguaje ensamblador con un compilador personalizado, que incluye una variante de C llamada 'Holy C' para los programas que corren en el sistema. Su diseño no sigue ningún estándar moderno de los sistemas operativos y tiene un enfoque *religioso* para representar el Templo de Salomón de la biblia, todo a raíz de una epifanía que Terry A. Davis tuvo.

Por lo general sólo se habla de la historia de TempleOS y su creador, un ingeniero que desarrolló trastornos mentales que encontró refugio en el desarrollo del 'sistema operativo de dios'. Pero para esta exposición, nos enfocaremos en qué son los anillos de protección, cómo es que TempleOS opera en el anillo 0, y las implicaciones que esto tiene en el manejo de procesos y la administración de la memoria.

Un poco de contexto

Terry A. Davis (15 Dic 1969 - 11 Ago 2018) fue un ingeniero eléctrico y programador de Estados Unidos. Obtuvo su título en la Universidad de Arizona, y ejerció como programador en *TicketMaster* hasta 1996, cuando empezó a tener ataques psicóticos. Poco tiempo después fue diagnosticado con trastorno bipolar y esquizofrenia.



Terry A. Davis trabajando en hardware para Ticketmaster

Originalmente empezó a desarrollar J Operating System en el 2005 para modernizar el sistema Commodore64, pero conforme pasaron los años y su estado mental se iba deteriorando, le cambió el nombre varias veces y le dió otro enfoque a su proyecto. TempleOS fue dado a conocer en 2014 tras casi 10 años de desarrollo; fue duramente criticado por sus capacidades (y porque en esos 10 años Terry se dedicó a hacer spam a través de todo el internet sobre su SO): resolución gráfica de 640x480 y solo 16 colores, una versión alternativa de C que no tiene nada importante que ofrecer, sin soporte para redes de ningún tipo, *todo* corre a nivel de kernel, el –único– usuario tiene acceso y control total sobre el hardware de la computadora en todo momento, no hace uso de memoria virtual, sólo física, y con la instalación se incluyen varios 'juegos' y programas para dar enseñanzas bíblicas y/o *hablar con dios*.

El mundo antes de los sistemas operativos multiusuario.

Antes de los sistemas operativos multiusuario, las computadoras eran monousuario y/o monotarea, diseñadas donde un solo usuario tenía control total de la máquina, sin separación entre niveles de privilegios, sin protección de memoria ni multitarea real. Este modelo era común en las décadas de 1950 a 1970, cuando los recursos computacionales eran extremadamente limitados y costosos.

Las primeras computadoras requerían que los usuarios cargaran programas manualmente, usando tarjetas perforadas o cintas magnéticas. No había distinción entre el sistema y el usuario: el usuario tenía acceso completo al hardware, es decir:

- No había protección frente a errores del usuario.
- No se podían ejecutar múltiples programas simultáneamente.
- Un fallo podría colapsar todo el sistema.

Con el paso del tiempo, y el aumento en la potencia de cómputo, se hizo necesario compartir los recursos entre múltiples usuarios de forma segura y eficiente. De ahí surgieron los sistemas multiusuario y multitarea, como UNIX en 1969, que introdujeron conceptos clave como:

- Separación entre usuario y kernel
- Gestión de procesos concurrentes.
- Sistemas de archivos seguros y jerárquicos.

- Control de acceso y permisos.

Y sin embargo, aunque TempleOS apareció en la segunda década de los 2000, este revive ese estilo anterior de computadoras; es un sistema simple, directo, sin capas de seguridad ni abstracción, concebido como una "modernización del Commodore 64" sin usuarios, sin redes, sin privilegios separados, haciéndolo tan vulnerable como limitado desde el punto de vista de la informática moderna.

Los anillos

En arquitecturas x86, los anillos de protección de un sistema operativo (o dominios de protección jerárquica) son un mecanismo de seguridad y aislamiento, que permiten proteger los componentes críticos de un sistema operativo. Son implementados a nivel de hardware, pero su implementación es completamente para el software. Organiza los niveles de acceso y privilegios en una estructura jerárquica de anillos concéntricos, donde cada uno define qué operaciones pueden realizarse y qué recursos están permitidos.

El anillo central, anillo cero o núcleo, tiene más privilegios que los anillos externos, ya que interactúa directamente con el hardware (como la CPU y la memoria del sistema), dirige la gestión de la memoria, la planificación de procesos y las llamadas al sistema. Los sistemas monousuario trabajan exclusivamente sobre este anillo.

Por su parte, los anillos 1 y 2 son utilizados en sistemas modernos para los drivers y sus privilegios son intermedios, y finalmente el anillo 3 es el nivel con menos privilegios, sobre el cual se ejecutan las aplicaciones de usuario, sin acceder directamente al hardware ni a la memoria del kernel. Cabe mencionar que, aunque el hardware de una computadora tenga soporte para varios anillos, los sistemas actuales realmente solo usan los anillos 0 y 3,

Como opera en el anillo 0, TempleOS no tiene separación kernel/usuario, un error en un programa puede sobreescribir memoria del kernel, no hay protección contra el malware, por lo tanto cualquier código puede leer y escribir sin permiso, así como modificar el código de otros programas en memoria. Hay más sistemas que opera(ban) a este nivel, como DOS y sus variantes, pero estos eran así por las limitaciones que había cuando se crearon, mientras que TempleOS fue una **decisión de diseño**. ¿Qué implicaciones tiene esto?

Procesos y multitareas

TempleOS redefine su concepto de gestión de procesos y multitarea; las distinciones entre procesos, hilos y tareas no existen, todos son llamados tareas, las cuales comparten un único espacio de direcciones de memoria sin protección ni virtualización de ningún tipo. Es multinúcleo, pero implementa un modelo en donde solo el núcleo principal *Core0* puede manejar procesos con interfaz gráfica, mientras que el resto actúan como “esclavos” limitados a ejecutar tareas en segundo plano.

El sistema gira entorno a dos procesos fundamentales: *Adam*, el proceso padre, se crea al iniciar el sistema, es indestructible (no se puede terminar de ninguna forma) y funge como un símil al *kernel* en otros sistemas; y *Seth* que maneja los procesos auxiliares presentes en cada núcleo del CPU y también es indestructible. Este se encarga de la creación de procesos hijos mediante funciones como Spawn() o PopUp(), y cuando un proceso termina, todos sus hijos son automáticamente eliminados para evitar fugas de recursos.

Gestión de memoria

TempleOS emplea un modelo de memoria sumamente sencillo y sin capas de abstracción: no hay memoria virtual, lo que significa que cualquier tarea puede leer o escribir en cualquier dirección de memoria, incluyendo aquellas reservadas para el sistema. La paginación está implementada (porque es requerida por la arquitectura de 64 bits), pero no se usa de ninguna forma. Para mantener eficiencia y simplicidad, el código está restringido a los primeros 2 GB de memoria —conocidos como "code heap"— lo que permite generar binarios compactos y facilita el uso de saltos relativos de 32 bits de una sola instrucción. Cada tarea gestiona su propio heap, que se libera automáticamente al finalizar la ejecución, mientras que el proceso "Adam" permanece como núcleo persistente del sistema. En cuanto a la interacción con el hardware, TempleOS accede directamente a través de técnicas como Memory-Mapped I/O (MMIO) y Port-Mapped I/O (PMIO), utilizando direcciones físicas y operaciones especiales del procesador, con un alias sin caché para evitar inconsistencias.

Seguridad

La seguridad en TempleOS es prácticamente inexistente por diseño, ya que todo el sistema opera en el anillo de máxima privilegio (anillo 0). Esta decisión permite máxima flexibilidad y rendimiento, pero a costa de estabilidad y protección: cualquier aplicación puede modificar la memoria del kernel, acceder a hardware sin restricciones o corromper estructuras críticas

del sistema con simples operaciones de escritura de memoria. No hay mecanismos de aislamiento, protección de memoria o control de acceso, haciendo que el sistema sea extremadamente vulnerable a errores o código malicioso. Este diseño refleja una simplicidad extrema y control directo sobre el hardware, sacrificando todas las protecciones que caracterizan a los sistemas operativos modernos a cambio de un rendimiento crudo y una transparencia absoluta en la operación del sistema. Su única protección es la falta de soporte para redes, pues una computadora con TempleOS estará completamente aislada del mundo exterior, y el único riesgo potencial será el propio usuario del sistema

Conclusión

La elección de TempleOS, fue por lo que representa, un sistema totalmente distinto a los que se usan hoy en día, aunque bien se pudo elegir MS-DOS o Commodore 64, también nos vimos influenciados por el hecho de lo que llevó a Terry diseñar este sistema, que se puede resumir a ser el tercer templo de Dios, de igual manera, lo que resalta es que su diseño rechaza los principios de seguridad y abstracción de los sistemas modernos, dando un control absoluto al usuario.

Nos deja pensando en lo que valoramos hoy: aislamiento, seguridad, eficiencia… pero también lo que hemos dejado atrás: el poder bruto, la transparencia, y esa cercanía directa con el hardware que solo un sistema como TempleOS puede ofrecer.

Referencias Bibliográficas

1. Božović, D. M. (2024). *TempleOS: Architecture and principles of lightweight operating system development* (Tesis de grado, Universidad de Kragujevac). ResearchGate. https://www.researchgate.net/publication/383849313_TempleOS_architecture_and_principles_of_lightweight_operating_system_development
2. **NDH Films – TempleOS Basics**
   NDH Films. (s. f.). *TempleOS Basics*. NDH Films. https://www.ndhfilms.com/other/templeosbasics
3. **Xe Iaso – TempleOS installation and basic use**
   Iaso, X. (2019, mayo 20). *TempleOS: Installation and basic use*. https://xeiaso.net/blog/templeos-1-installation-and-basic-use-2019-05-20/
4. **Minexew – TempleOS loader (parte 2)**
   Minexew. (2020, marzo 29). *TempleOS loader – Part 2*. https://minexew.github.io/2020/03/29/templeos-loader-part2.html
5. Sari S. (2024) *What are rings in operating systems?*. Baeldung. https://www.baeldung.com/cs/os-rings
6. TutosPC (2025) *El SISTEMA OPERATIVO creado por un Genio con Esquizofrenia | TempleOS* [Video] Youtube https://www.youtube.com/watch?v=-q_pcKroyJg&t=350s
7. Dodgie (2024) *El BIZARRO programador que HABLABA CON DIOS* [Video] Youtube https://youtu.be/Mu-ihaH4QxI?si=HszSWWLYIGk9_Uoa