

TEMPLE OS

***LAS IMPLICACIONES DE
UN SO OPERANDO EN EL
ANILLO O DE PROTECCIÓN.***

Juarez Valdivia Barvara Caridad
Talonia Fuentes Jesús



¿Qué es TempleOS?



TempleOS es un SO libre desarrollado por Terry A. Davis. Es un sistema de código abierto y de arquitectura de 64 bits, "famoso" por su enfoque bíblico y unas cuestionables decisiones de diseño.

Es un sistema **monolítico y monousuario** de 64 bits, con soporte para apenas 16 colores en pantalla, resolución de 480x640p, un solo canal de audio y que corre una versión personalizada del lenguaje C, llamada 'Holy C'

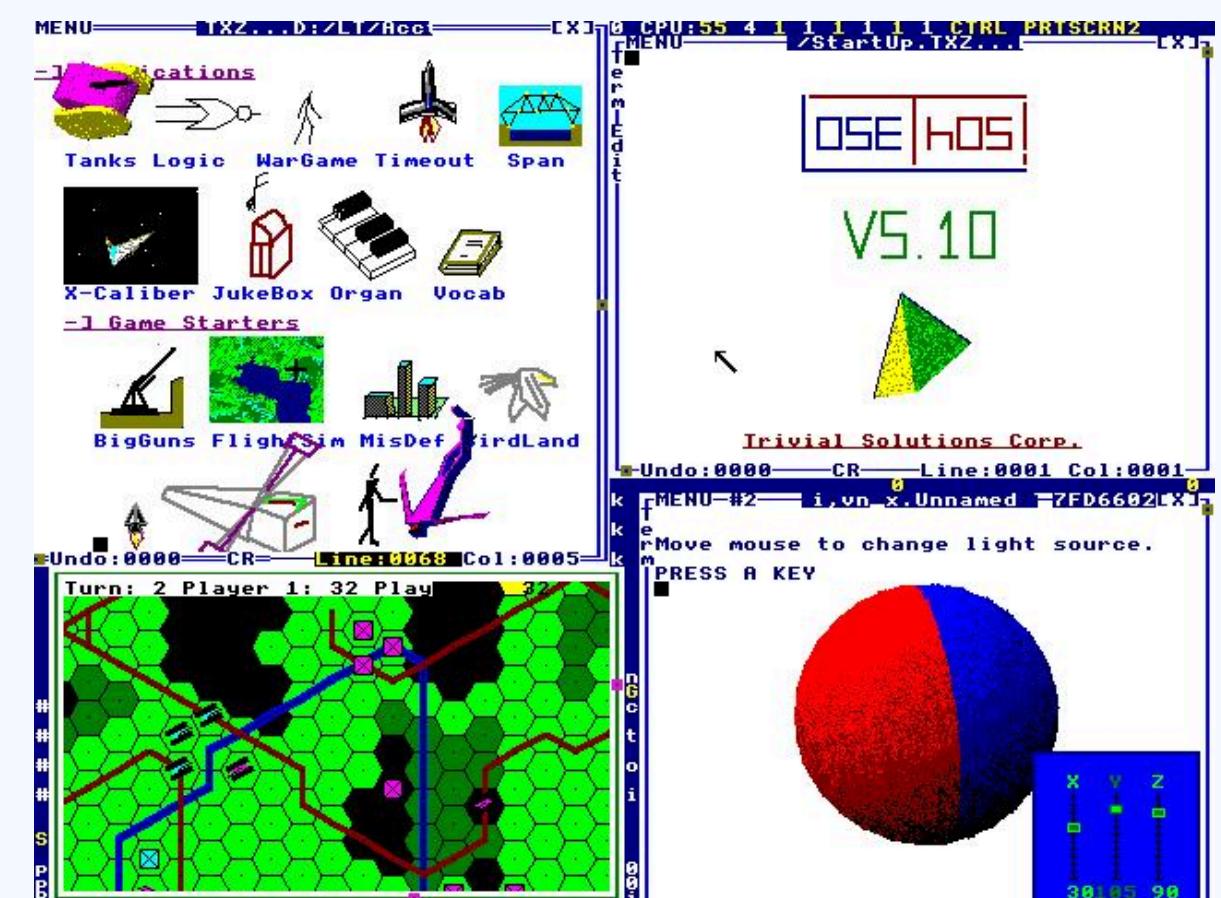
Un poco de contexto

Terry, su creador, fue un ingeniero eléctrico y programador estadounidense que sufría del trastorno de personalidad múltiple, bipolaridad y esquizofrenia. Empezó el desarrollo de TempleOS a partir de una epifanía en la que Dios mismo le pidió que le construyera un sistema operativo

Originalmente era un proyecto personal de Terry para modernizar Commodore64, pero al perder su empleo se dedicó enteramente a su desarrollo

Originalmente se llamaba **J Operating System**, y al pasar los años fue renombrado como **SparrowOS**, **LoseThos** y finalmente **TempleOS** en su versión final.

Comunmente se habla de TempleOS y la historia de su creador, pero para esta exposición nos centraremos en cómo opera este sistema de 'Arquitectura moderna' pero con un diseño antiguo



SO's monousuario

Los primeros sistemas operativos como MS-DOS no tenían una distinción entre lo que hace el sistema y lo que hace un usuario; toda tarea o proceso tenía acceso a todos los recursos de hardware sin restricciones. Esto conlleva:

- No existe protección de memoria
- Todo corre con los privilegios máximos del sistema
- El usuario tiene acceso directo a cualquier puerto y dirección de memoria
- No hay multitarea real
- El SO es más sencillo y compacto por la carencia de estas características
- A cambio, es muy vulnerable a fallos de integridad

Estas características eran dadas por las limitaciones que tenía la computación entre 1950 y 1970. Con el tiempo, se hizo necesario compartir recursos entre múltiples usuarios de forma segura y eficiente. De ahí surgieron los sistemas multiusuario y multitarea. Una de las principales ideas para esto, fue la implementación de **anillos de sistema**

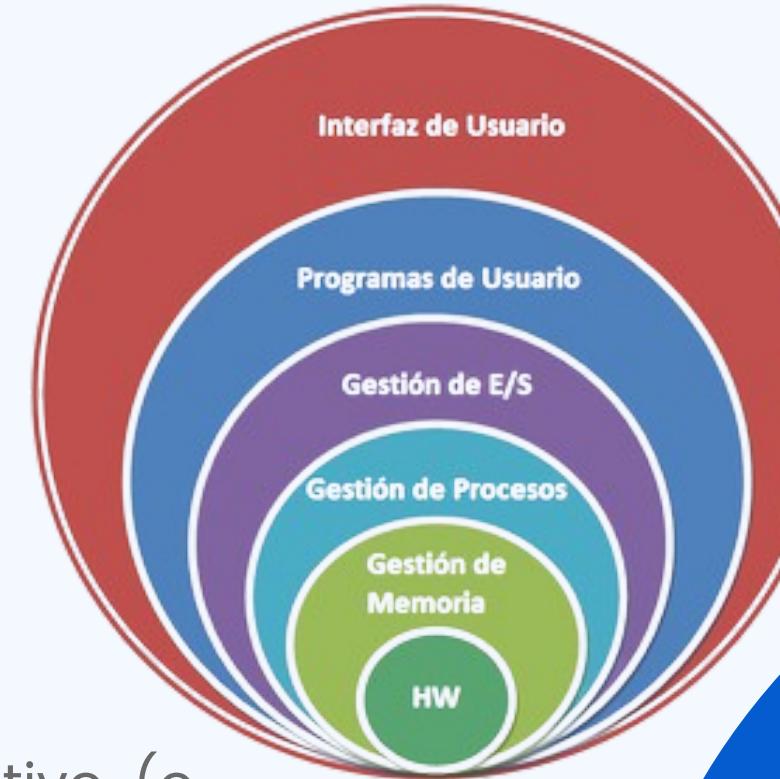


Anillos de Protección

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

Es una estructura a nivel de hardware y aprovechada por el software, que limita los accesos y privilegios en el CPU, representada con anillos concéntricos. El nivel de privilegio empieza en el anillo 0, y disminuye en cada anillo exterior

Se trata de una implementación lógica y funcional de la microarquitectura de los procesadores. Por lo que, aunque se trata de hardware, no es algo que físicamente se pueda apreciar, sino que son parte del conjunto de instrucciones y el modo de operación interna del procesador



ANILLO 0

El anillo central, cero o núcleo, interactúa directamente con el hardware; en los sistemas modernos es aquí donde se manejan todas las funciones de un sistema operativo, como las llamadas al sistema, la planificación de procesos y la administración de memoria

ANILLO 3

Usualmente es a partir de este anillo que se realizan las tareas del usuario, como la ejecución de aplicaciones y las funciones relacionadas a la interfaz gráfica

ANILLOS 1,2, Y MÁS

Los anillos entre el nivel del kernel y el nivel de usuario son usados para otras funciones que no requieren necesariamente del mayor nivel de privilegios (como la gestión de los drivers y E/S), pero que no se le pueden delegar al usuario.

Aunque los procesadores tengan soporte para estos y más anillos (y estos se mencionen en la documentación), en la práctica sólo se utilizan los anillos 0 y 3.

La separación entre anillos del kernel del SO y los usuarios son un estándar común desde los 80's. Y aún así, TempleOS vive completamente en el anillo 0 por diseño, para que fuese *más sencillo* y *flexible* ¿Qué implicaciones tiene esta decisión?

Procesos y Multitareas

Para TempleOS no hay diferencia alguna entre procesos y tareas, todos son llamados tareas, y comparten el mismo espacio de direcciones.

Se usa un modelo "Amo-Esclavo", en donde un proceso **Adam** se adueña del Core0 (núcleo principal) para ser ell "amo", mientras que los demás núcleos corren tareas "esclavos" en segundo plano bajo el mando del proceso **Seth**. Por ello, la multitarea en TempleOS es cooperativa, no preventiva.

Las tareas se generan con las funciones `Spawn()` y `PopUp()`, que son un equivalente a `fork()` de UNIX, pero escritas en Holy C

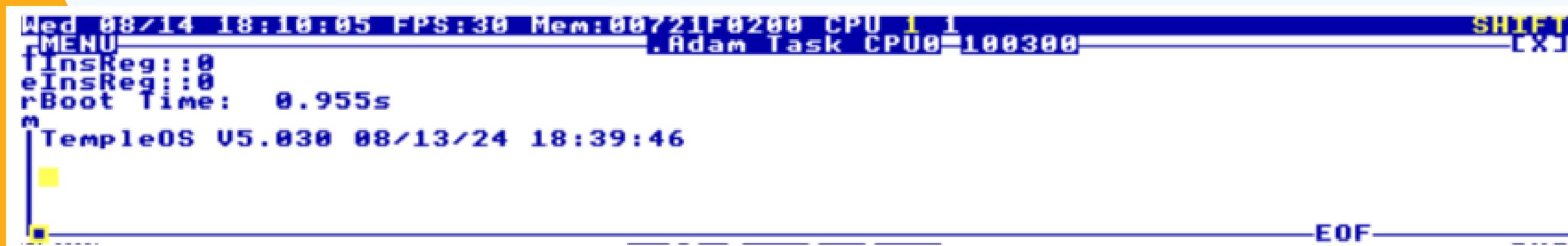
Adam y Seth

Adam - Padre de todos los procesos e indestructible.
Seth - maneja los procesos auxiliares en cada núcleo de la CPU.

Core0

Es el nucleo principal del sistema. Sólo este puede mostrar ventanas o ejecutar aplicaciones con interfaz.

La comunicación entre núcleos es manejada por Adam y Seth, pero no existe ningún mecanismo de sincronización. Las tareas se bloquean constantemente, pero sólo se bloquean las manejadas por Seth en favor de lo que Adam esté haciendo



Gestión de memoria

No hay memoria virtual, las direcciones virtuales y físicas están mapeadas de forma idéntica.

No hay paginación, las direcciones se usan de forma directa e idéntica para todos los procesos.

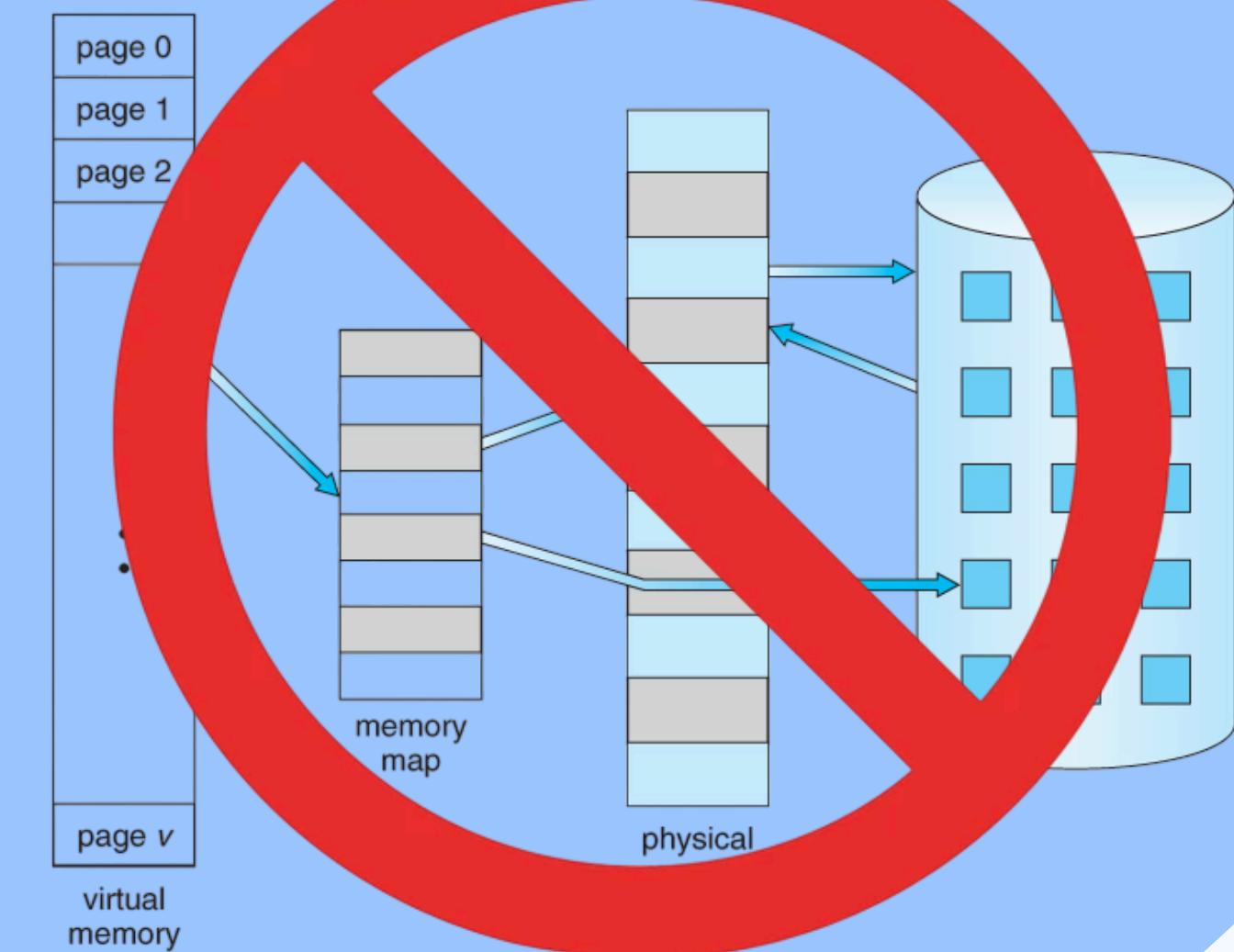
La memoria asignada a una tarea se libera automáticamente al finalizar para evitar fugas de memoria.

Solo se permite código de programa en los primeros 2GB de memoria, para permitir binarios más compactos que funcionen con instrucciones CALL y JMP de 32 bits

TempleOS trabaja con acceso directo a registros de hardware usando direcciones de memoria físicas, por lo que no utiliza drivers externos ni módulos de kernel para los puertos de E/S. Todo el acceso a hardware (como teclado, mouse, disco, video) se hace directamente en HolyC, el lenguaje nativo del sistema, mediante instrucciones como **inb**, **outb**, **poke**, **peek**, etc.

Code heap

Ocupa los primeros 2GB de RAM y contiene el código. Se pueden crear heaps personalizados.



Operaciones de entrada y salida

TOS utiliza 2 rutinas para administrar sus (muy pocos) dispositivos de E/S:

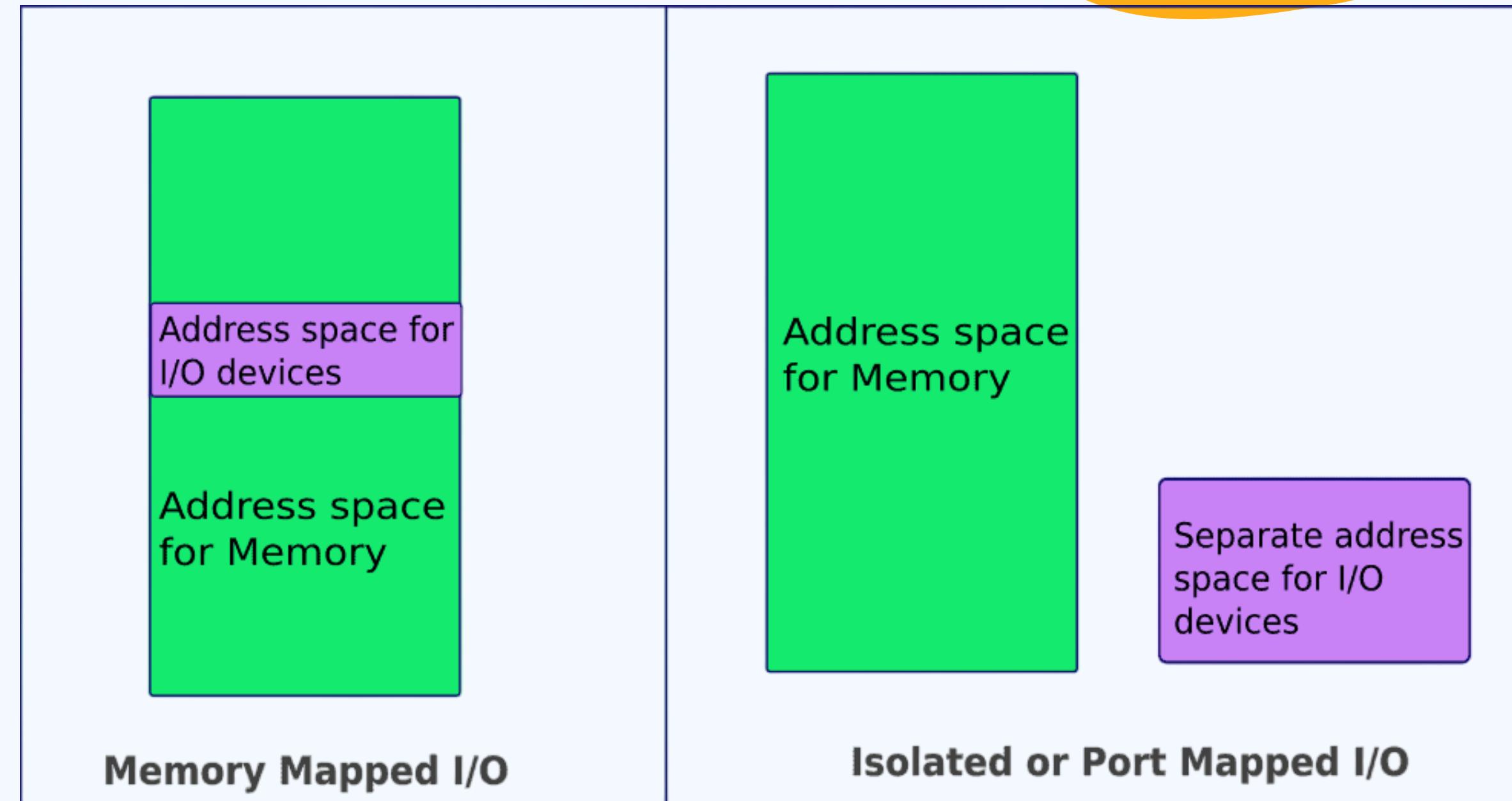
MMIO (Memory-Mapped I/O)

Permite el acceso directo a los registros de hardware usando sus direcciones de memoria físicas.

PMIO (Port-Mapped I/O)

Uso de instrucciones específicas de CPU *in* y *out* para leer y escribir en puertos de hardware.

- La E/S esta unificada con el espacio de memoria.
- Se usan alias sin caché para evitar lecturas de datos antiguos al interactuar con el hardware.
- Funciones como `FileRead` y `FileWrite` permiten realizar operaciones de E/S de manera eficiente y directa con el código de Holy C



Seguridad

En TempleOS no existe ninguna medida de seguridad *planeada*. 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 (según Terry) 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



**TempleOS Boot Loader**

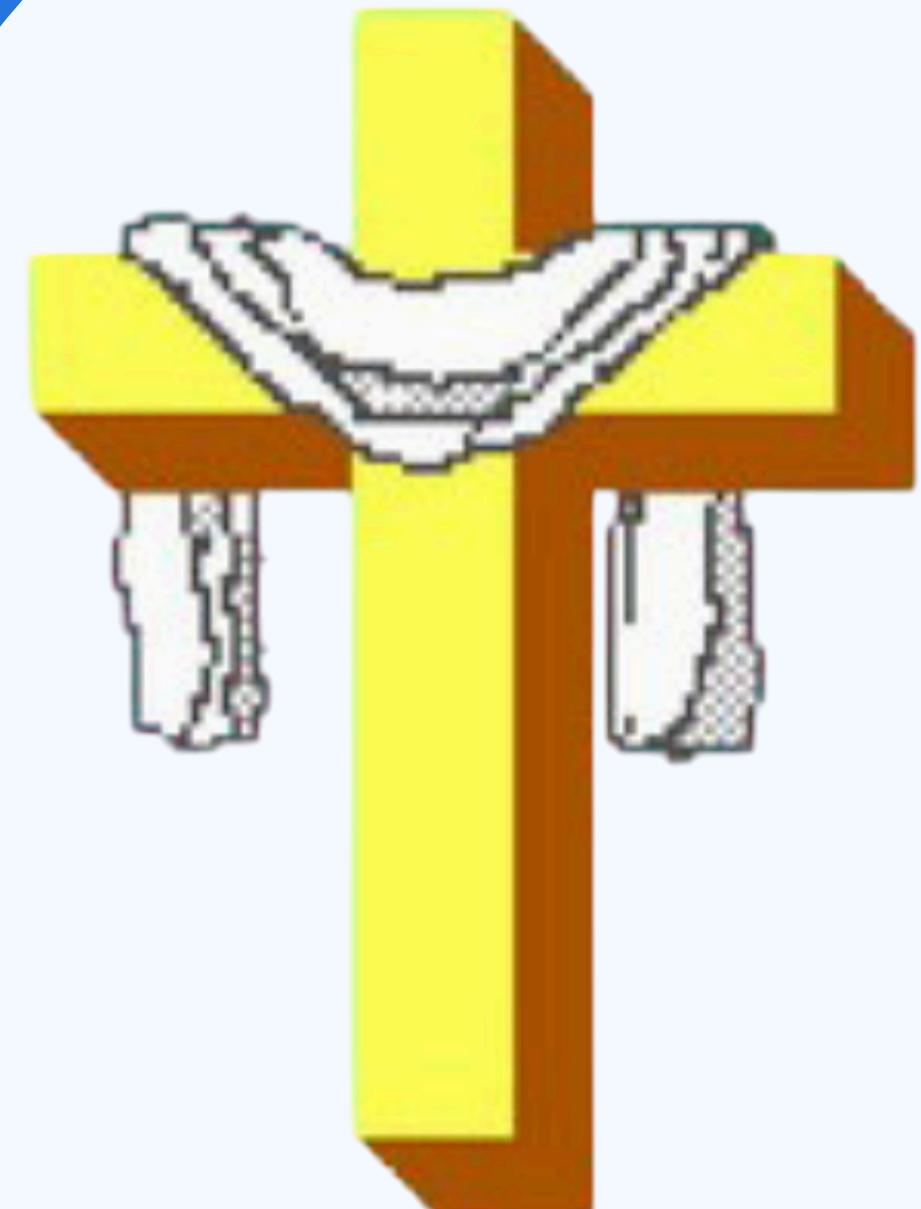
- 0. Old Boot Record
- 1. Drive C
- 2. Drive D

Selection: _

Conclusiones

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.



MUCHAS GRACIAS

Referencias Bibliográficas:

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