



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería

Sistemas Operativos

TempleOS

Alumno:

Hernandez Gallardo Daniel Alonso

Profesor:
Gunnar Wolf
Grupo 6, Semestre 2025-1

September, 2024

Índice general

| | |
|--------------------------------|----|
| 1. Introducción | 2 |
| 2. ¿Quien fue Terry A. Davis? | 3 |
| 3. TempleOS | 4 |
| 3.1. Instalación | 4 |
| 3.2. ¿Como funciona? | 4 |
| 3.2.1. Shell | 6 |
| 3.2.2. HyperText | 8 |
| 4. HolyC | 9 |
| 5. Conclusiones | 10 |

1 Introducción

Los sistemas operativos siempre se han mantenido en un margen muy comercial, pues su propósito es brindar al usuario el poder de interactuar con el hardware que tienen a su disposición, algunos sistemas operativos suelen ser más liberales que otros; Windows en su caso es muy útil para el usuario menos técnico, pues esta construido en base que le resulte todo obvio al consumidor y tenga la menor probabilidad de echarlo todo a perder, Linux es más complejo, siendo así menos restrictivo, es más liberal y personalizable pero el usuario tendrá que enfrentarse a muchas más cosas de lo usual y al final tenemos un sistema operativo como TempleOS, algo que no esta planeado para nada con un uso clientelar o experto, un sistema operativo ligero con acceso total, un sistema operativo que trabaja en el anillo 0 teniendo acceso total al hardware en cualquier lugar que lo trabajes.

El creador único de este sistema operativo (y de quien indagaremos más adelante). Terry A. Davis, hace la siguiente analogía:

- Linux es como un camion con 20 marchas para operar.
- Windows es como un coche
- TempleOS es como una motocicleta, donde si te inclinas demasiado puedes caerte.

Tratando de decir que las limitaciones de este sistema operativo son tan pocas, que con cualquier cosa que te salgas de los parámetros establecidos por el mismo este caerá.

2 ¿Quien fue Terry A. Davis?

Terry A. Davis fue a quien considerarías un prodigo en la programación, fue capaz de crear su propio sistema operativo (Temple OS) el solo, además de usar su propio lenguaje de programación llamado Holy C (Lo que seria como un C con esteroides). Terry A. Davis lastimosamente falleció en el ya no tan reciente año de 2018. Él a pesar de lo inteligente y prodigo que pudo llegar a ser sufría de esquizofrenia, lo que lo llevo a un declive en su salud y que lo hizo entrar en muchas polémicas pero quiero mencionar esto solo por lo siguiente que quiero citar:

“Espero que Davis sea recordado por sus logros y no por su enfermedad mental” – John McColl, Cecil, 2018.

Una vez dicho esto y haber mencionado un poco sobre la vida de Terry A. Davis podemos hablar de sus logros.



Figura 2.1: Terry A. Davis

Extraido de: <https://www.disabilitysupportguide.com.au/talking-disability/bullied-schizophrenic-genius-the-sad-story-of-terry-davis> McManus, 2023

3 TempleOS

TempleOS fue el sistema operativo creado por Terry y únicamente por él, según Terry, este sistema operativo nace porque Dios lo instruyó para construir un templo (de ahí el nombre), el tercer templo de dios para ser específico; muchas veces por esto la gente tiende a restarle importancia a este sistema operativo, pero este no es el caso, pues para mí la verdad es un hito que un solo hombre haya podido crear su propio sistema operativo, aunque este haya sido “según las ordenes de dios”, es interesante ver un sistema operativo totalmente personal y que no tenga que seguir estándares o tenga que apegarse al mercado de las computadoras, además a pesar de que el sistema operativo tenga un sentido “divino”, en realidad es el trabajo de 12 años de un hombre, pues nació como “J operating System”, que pasó a ser LoseThos, que pasó a ser SparrowOS, hasta terminar en TempleOS. Así que no solo eran los delirios de un hombre enfermo, era algo más que eso.

“Según su creador TempleOS es “un sistema operativo x86-64, multirata, multinúcleo, de dominio público, open source, Ring-0-only, de un solo espacio de direcciones (mapeo de identidad), sin conexión y para programación recreacional.” – Butler, 2024

Lo sé suena muy complicado pero a lo largo de este texto iremos explorando cada una de sus partes.

3.1 Instalación

TempleOS era un sistema operativo que estaba diseñado para programar por lo que era necesario que fuera lo más rápido posible, esto hace que su instalación sea realmente muy rápida, arranca en el disco duro en un segundo, no usa paginación por lo que todo el sistema se carga directamente en la memoria ram, ¿puedes creerlo?, un sistema operativo que puedes usar al instante, imagina que pudieras cambiar entre una distribución de linux y windows en un segundo, sería algo muy útil e incluso abre la puerta a tener un sistema operativo dual. TempleOS se puede adquirir haciendo click aquí, y se puede arrancar en una máquina virtual.

3.2 ¿Cómo funciona?

TempleOS como menciono su creador, trabaja en el anillo cero (Ring-0-only), por lo que todo se maneja desde el propio núcleo del sistema sin tener ningún tipo de seguridad, tiene acceso total al hardware. Esto lo hizo porque para él la protección solo es algo que ralentiza las cosas y hace el código complicado. También

el hecho de que sea un sistema operativo de un solo espacio de direcciones trae varias implicaciones:

1. Todas las aplicaciones y procesos comparten un único espacio de direcciones así que varias partes del sistema con distintos niveles de privilegio pueden acceder a la misma región de memoria, es decir, la memoria no tiene protección alguna.
2. Hay una correspondencia clara y única entre las direcciones que usan los programas y las direcciones que usa el hardware por lo que no hay que hacer una transformación compleja entre ellas durante el acceso a memoria, mejorando así el rendimiento.

La principal ventaja de usar un sistema operativo de este estilo es que puede usarse para mejorar la estructura y rendimiento de las aplicaciones como del propio sistema operativo, pues reduce la necesidad de comunicación explícita entre programas que se ejecutan en niveles de protección distintos.

El sistema operativo se basaba totalmente en darle todo el poder al único usuario que podía usarlo, por esto es que no posee algo como permisos de acceso; el sistema tampoco usaba hilos pues se puede acceder a la memoria sin ninguna restricción, solo haz otro proceso y listo.

Otras características

°El sistema corre en ASCII de 8 bits, con un sistema de gráficos incorporado directamente en el código fuente; Temple OS trabaja a un máximo de 640x480 VGA con 16 colores que según Davis, Dios le dijo específicamente estas características para que los niños pudieran leer la pantalla y hacer mejores dibujos para dios.

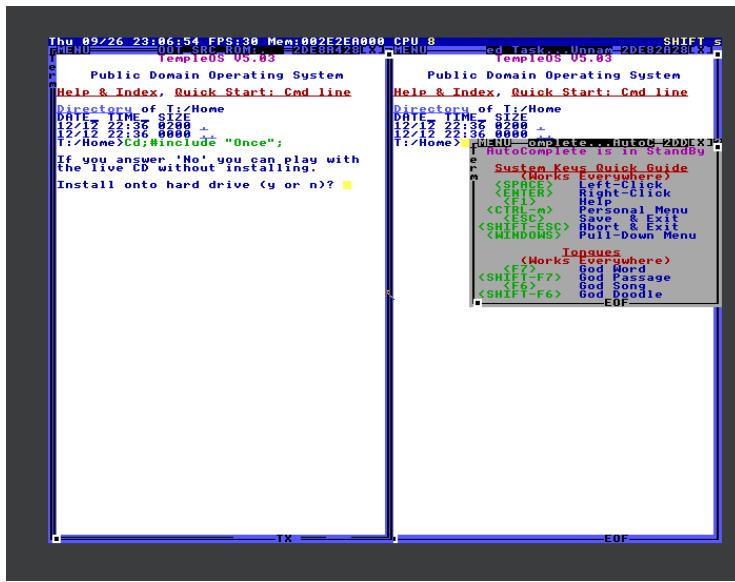


Figura 3.1: Temple OS

Un diseño bastante llamativo, pero supongo que el no hubiera sido muy fan de la tendencia actual del modo oscuro.

3.2.1 Shell

El shell de TempleOS usa su propio lenguaje de programación, este fue bautizado como HolyC, de hecho, todo el sistema esta programado (a excepción del entorno ensamblador) en este lenguaje, el lenguaje es claramente muy similar a C por lo que puedes usar el shell como si programaras en C; por esto mismo es que no hay una aplicación de calculadora en el sistema operativo, si quieres hacer cálculos simplemente se lo pides al shell.

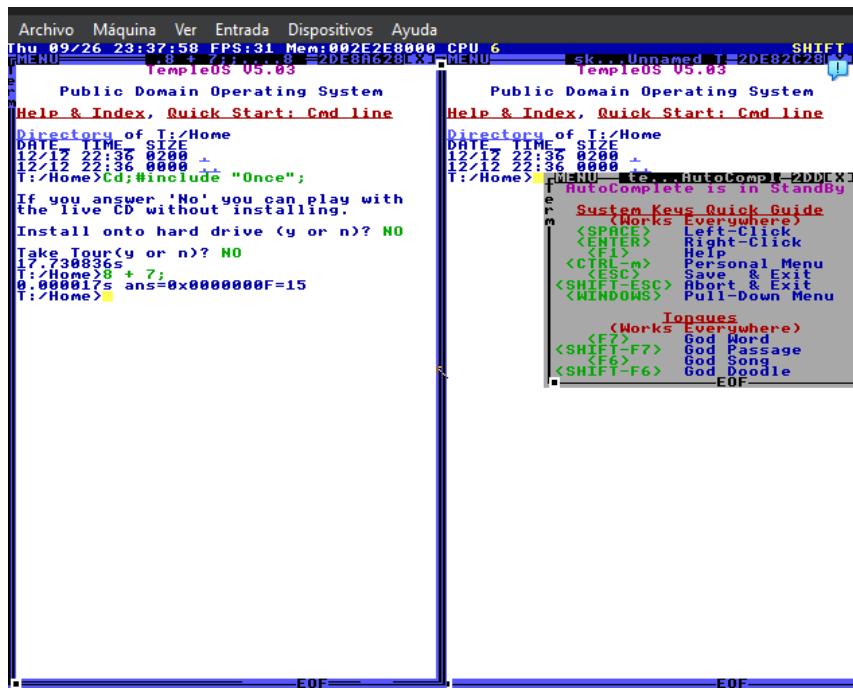


Figura 3.2: Calculo en el Shell

El shell se comporta de manera similar al Interactive Shell de Python o a la linea de comandos de otros lenguajes interpretes como Ruby o Lisp.

Explorador de archivos

El explorador de archivos es propio de TempleOS y denominado RedSea, supondré que porque “navegas” en directorios de color rojo como se muestra en la siguiente imagen:

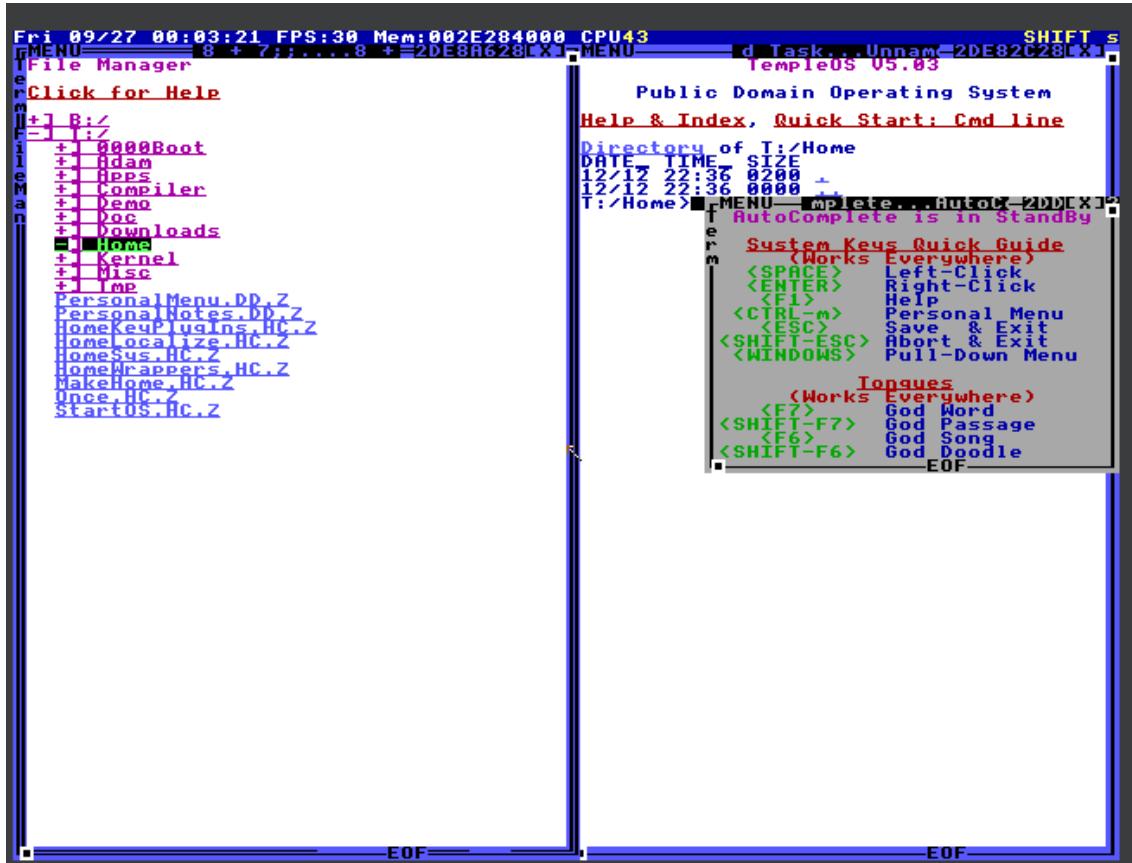


Figura 3.3: Administrador de archivos

Pero en realidad es más una extensión del shell, pues con el sistema de hipertext con el que trabaja TempleOS, el shell puede actuar fácilmente como un explorador de archivos.

```
Take Tour(y or n)? NO
3.611208s
T:/Home>Dir;
Directory of T:/Home
DATE    TIME   SIZE
12/12  22:36  0200  -
12/12  22:36  0000  +
3.001099s ans=0x00000002=2
T:/Home>
```

Figura 3.4: Usando el shell como explorador de archivos

3.2.2 HyperText

El hypertext es un sistema que permite enlazar documentos de forma que puedes crear conexiones interactivas entre diferentes partes de información (Algo así como cuando en el índice de este texto haces clic en algún capítulo y te redirige hacia el o si haces clic AQUI te redirijo a mi capítulo favorito de este texto), este sistema está implementado a través de DolDoc. La cosa es que con DolDoc no solo podías hacerlo con texto, sino con casi cualquier cosa. Esto es bastante interesante en realidad porque podías poner hipertexto incluso en donde estés programando por ejemplo en una función para que te mande a su diagrama de flujo. Por esta razón el sistema operativo no venía con un programa para dibujar, presionar CTRL+R podías hacer lo que se te diera la gana, referenciarlo y pegarlo en otro lado cuando quisieras.

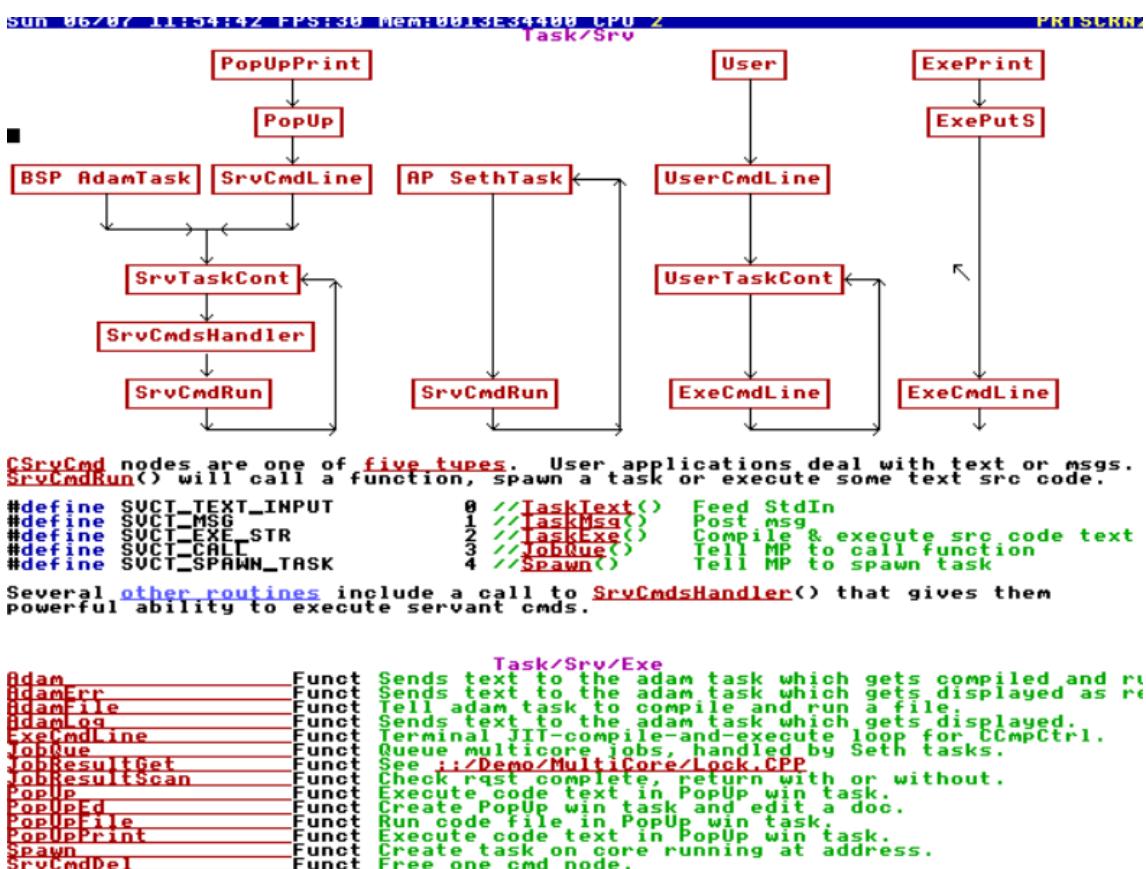


Figura 3.5: Ejemplo de un diagrama de flujo que usa HyperText; extraido de: <http://www.codersnotes.com/notes/a-constructive-look-at-templeos/> Mitton, 2015

“Y si, todas esas cajas del diagrama de flujo son hipervínculos, así que puedes hacer clic en ellos para ir directamente al código fuente que lo implementa” – Mitton, 2015

4 HolyC

Ya mencionamos a HolyC antes en este texto, pues es de vital importancia considerando que prácticamente todo el sistema operativo esta basado en el, incluso el shell se usa con base a HolyC; HolyC es un C expandido, tal vez no al nivel de c++ pero si tiene una cantidad razonable de extensiones. No hay función main() pues todo TempleOS (sin contar el kernel o el compilador) es Just-in-Time y se compila cuando se requiera, por lo que solo usando el comando “#include” puedes compilar donde necesites, puede ser desde una fracción de código hasta una función entera. Algo que es muy interesante es la directiva #exe, que te permite acceder a cosas que están fuera de tu código y usarlas a tu beneficio.

Algunas características que lo hacen interesante son:

1. Los meta datos de los miembros de las clases de Holy C pueden ser asignados en el tiempo de compilación.
2. La palabra reservada lastclass se puede utilizar para dar un argumento por defecto a las funciones, haciendo que si una función no recibe un argumento proporcionara el nombre del tipo del argumento anterior como una cadena de texto.
3. No genera archivos objeto.
4. Usa un enlazador dinámico que solo vincula los símbolos en el proceso de carga por lo que la tabla de símbolos es accesible incluso en la ejecución del programa.
5. No hay variables de entorno

La verdad es que este lenguaje de programación implementa muchas libertades que incluso C++ podría envidiar, cosas como que el debugger puede funcionar verificando linea por linea del código, puedes ejecutar secciones específicas del mismo, en cierta forma lo miro como una combinación entre C y Python.

5 Conclusiones

Quiero concluir esto citando a alguien, pues su reflexión me pareció muy interesante:

“En un video Terry muestra como construir una aplicación gráfica desde cero y se puede apreciar lo sencillo que era, para replicar esto en windows seria necesario una serie de pasos extensa (registrar clase de ventana, crear ventana, ejecutar comandos de interfaz de dispositivo grafico (GDI por sus siglas en inglés), tener una cola de mensajes y más), configurar el proyecto en Visual Studio, incrustar un bitmap o cargarlo desde el disco, mientras que Terry solo usa un pequeño fragmento de código. Definitivamente te hace preguntarte en que momento nos desviamos tanto”
–Mitton, 2015

Y esto es muy cierto, realmente en este sistema operativo y por todas las herramientas que ofrece se hace visible que esta hecho con la intención de que programar sea fácil, instalarlo es fácil, acceder a la memoria es fácil, acceder a tus archivos es fácil, acceder al hardware es fácil, realmente todo es muy sencillo porque esta planeado para serlo, porque esta pensado para que el sistema operativo sea usado por alguien que en definitiva sabe lo que hace con el y lo usa como herramienta para controlar su sistema, cuando en la actualidad el sistema operativo se ha vuelto la herramienta para “mantener” al sistema. Este ejemplo de la interfaz gráfica me hizo mucho sentido porque pase por lidiar con eso, en Java fue terrible lidiar con eso, tuve que usar programas externos, bibliotecas externas y todo para poder crear a mi gusto la interfaz gráfica y aun así quede insatisfecho, al final poder lograr correr ese programa termino siendo un infierno para mi como programador, pero al final para el usuario fueron simples clics, tal vez por eso la comunidad de programadores disfrute un poco más de esa libertad que te brinda Linux, y estoy seguro que Davis disfruto como nadie la libertad total de su sistema operativo. Todo esto en conjunto lo hace interesante, pues nos abre las puertas crear sistemas operativos con distintos enfoques, e ir más allá del enfoque de hacerlo sencillo para el usuario común, pues la meta de conseguir una computadora en cada hogar, pienso yo, ya ha sido alcanzada.

Sería genial encontrar más variedad de sistemas operativos, ahora mismo las computadoras se trabajan en Windows o distribuciones de Linux (no quiero mencionar Mac OS X porque cuando armas tu computadora piensas si comprar una licencia de Windows o que distribución de linux te beneficia más, pero si compras una Mac el sistema operativo es tu menor preocupación), pero ¿cuándo entrara al mercado una tercera opción?, ¿Estas dos opciones satisfacen tanto nuestras necesidades que se han convertido en un estándar?; otro sistema operativo del que me hubiera gustado

hablar y quienes, en una extraña coincidencia mi Padre, y mi Profesor me dieron a conocer es Fenix, creado por la familia Toledo, que cansados de las alternativas que tenían en el mercado (que en aquel tiempo eran Windows XP y Linux) decidieron crear su propio sistema operativo ajustado a su propio hardware, pues a ellos les parecía que:

“En la actualidad (2002), una instalación Linux requiere prácticamente igual o más recursos que Windows XP, con un mínimo de 128 MB. de memoria, al parecer ambos compiten por ver quien se hace más lento.”
–Toledo, 2002

Otro caso muy interesante, pues si en este texto vimos el caso de un hombre que creó su propio sistema operativo, imagina que tan interesante es que una familia entera cree sus computadoras, su sistema operativo y sus programas.

Bibliografía

- Butler, S. (2024). The strange story of TempleOS. *XDA Developers*. <https://www.xda-developers.com/strange-story-templeos/>
- Cassel, D. (2018). The troubled legacy of Terry Davis, 'God's Lonely Programmer' [Archived from the original on September 28, 2018. Retrieved September 28, 2018.]. *The New Stack*.
- Cecil, N. (2018). Man killed by train had tech following [Archived from the original on November 8, 2020. Retrieved November 24, 2020.]. *The Dalles Chronicle*.
- Hicks, J. (2014). God's Lonely Programmer [Consultado el 21 de abril de 2015]. *VICE Motherboard*. <https://www.vice.com/en/article/gods-lonely-programmer>
- Koldinger, E. J., Chase, J. S., & Eggers, S. J. (1992). *Architectural support for single address space operating systems* (inf. téc.). Department of Computer Science, University of Washington. Seattle, WA.
- McManus, D. (2023). Bullied schizophrenic genius: The sad story of Terry Davis [Posted 2023 year ago]. *Disability Support Guide*. <https://www.disabilitysupportguide.com.au/talking-disability/bullied-schizophrenic-genius-the-sad-story-of-terry-davis>
- Mitton, R. (2015, junio). A constructive look at TempleOS [I'm Richard Mitton, a freelancing British software engineer and part-time beard-grower, now based in Los Angeles.]. <http://www.codersnotes.com/notes/a-constructive-look-at-templeos/>
- Toledo, F. (2002, diciembre). Sistema Operativo Iberoamericano [NFT. 5 de diciembre del 2002]. <http://www.biyubi.com/art19.html>