UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

"SISTEMA ANTIVIRUS MULTIPLATAFORMA EN TIEMPO REAL USANDO TÉCNICAS HEURÍSTICAS Y PROACTIVAS - 2013"

TESIS

PRESENTADA POR:

Bach. RAMIRO PEDRO LAURA MURILLO

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

PUNO - PERÚ

2013

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO
FACULTAD DE INGENIERIA ESTADÍSTICA E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

SISTEMA ANTIVIRUS MULTIPLATAFORMA EN TIEMPO REAL USANDO TECNICAS HEURISTICAS Y PROACTIVAS - 2013

T E S I S

Presentada por:

Bach. RAMIRO PEDRO LAURA MURILLO

A la Coordinación de Investigación de la Facultad de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano – Puno, para optar el Titulo Profesional de: INGENIERO ESTADÍSTICO E INFORMÁTICO

APROBADA POR:

PRESIDENTE	:	M.Sc. ERNESTO NAYER TUMI FIGUEROA
PRIMER MIEMBRO	:	M.Sc. REMO CHOQUEJAHUA ACERO
SEGUNDO MIEMBRO	:	M.Sc. FREDY HERIC VILLASANTE SARAVIA
DIRECTOR	:	M.Sc. CESAR AUGUSTO LLUEN VALLEJOS
ASESOR	:	Dr. VLADIMIRO IBAÑEZ QUISPE

PUNO - PERÚ
2013

Área : **Informática**
Tema : **Ingeniería de software e inteligencia artificial**

DEDICATORIA

Al increíble equipo de **K.A.V.** *(Killer Antivirus 2003)*, quienes me enseñaron C/C++ avanzado, libros sobre Protected-Mode, Virus y muchísimo **Assembler**, fueron años increíbles de noches de desvelo frente a rudimentarias redes de computadores. Gracias a todo ese sacrificio pudimos obtener gran conocimiento sobre seguridad, criptografía y sistemas operativos.

AGRADECIMIENTOS

Quiero manifestar mi agradecimiento a todos mis profesores de la **Facultad de Ingeniería Estadística e Informática** a quienes he llegado a apreciar de gran manera gracias por su impecable labor, continuas enseñanzas y apoyo durante estos años de estudio y dedicación. En el plano académico mis memorias para el **Dr. Ernesto Cuadros**, un gran científico en Ciencias de la Computación incansable soporte de la Sociedad Peruana de Computación.

Quisiera también expresar mi gratitud a mis compañeros por su respaldo personal y amistad incondicional manifestada en el día a día. A una gran mujer y amiga, a mi recordada **Nancy**, aunque pasen las personas, estarás siempre en mi corazón. A mis primeros y recordados amigos de la universidad mi "viejo" *William, Leo, Dante, Richard, Anais, Techi, Jacky, Jeny* en las noches *"kankun"* y a mis siempre maravillosas amigas *Lin, Yojhis y Marilady*. En esta última etapa universitaria y de **nova vita** a los "gatos" *Sandrita, Ross, Lixita Romel, Mario, Perez, Wilson, Boris, Peyuko* y definitivamente a nuestra gran criatura el gran **FILICC**, el cual nació cerca del 2005 con mis cofundadores ***Vlady y Chizoz***, con el sitio web activo hasta el presente.

Manifestar mi admiración y futuros éxitos a mis "***ProgrammerPartners***" y buenos amigos ***Vlady, Herles, Joel, Asqui, Suaquita, Jallo, Julio, Sergio*** los *HeadMasters*, también a ***Jefry, Efra, Yosef y Rudy*** los *TargetMasters*.

Finalmente, lo mas importante, agradezco el apoyo incondicional de mis queridas y singulares "hemanitas" ***Marivel y Rocio***, como a a mis comprensivos y dedicados padres ***Pedro y Adela***.

Mencionarles también que la vida es un gran libro, no importa el tiempo que te tome escribirlo, importa que contiene, importa la participación de los actores, los amigos que uno pueda hacer participe de su vida, la forma peculiar en la que cada uno influye sobre ti, siempre es necesario saber escuchar y reconfortar a otros, se aprende de tantas personas y edades, cada hoja de nuestra vida podrá reconfortarnos y minimizar la depresión, nos alimentamos de emociones compartidas, es necesario saber escoger lo adecuado y nosotros ser adecuados para la vida.

A todos, muchísimas gracias.
Ramiro Pedro Laura Murillo.

RESUMEN

Se tiene como problemática la proliferación y propagación de una gran variedad de virus informáticos a través de los dispositivos de almacenamiento extraíbles como son: la contaminación lógica del sistema operativo, la destrucción de archivos y carpetas, la posibilidad de daños en los medios de almacenamiento y otros. La presente investigación tuvo como objetivo "Desarrollar un Sistema Antivirus con detección en tiempo real usando técnicas Heurísticas, Proactivas y auto-actualizable", aprovechando la conexión a Internet. Cumpliendo con este objetivo de desarrollo el Sistema Antivirus de nombre AV-Fenix el cual realiza las operaciones de detestación, eliminación, prevención de software malicioso, actualización de la base de firmas y revisión de los dispositivos extraíbles que se conecten al computador.

La metodología que se usó fue de la Programación Extrema (XP) por su flexibilidad en el desarrollo de aplicaciones pequeñas y eficaces sobre todo por que el antivirus es modificado e incrementado con mejoras necesarias por la constante evolución de los malwares. El Modelamiento basado siempre en el Lenguaje de Modelamiento Unificado (UML) y la Métrica de Validación ISO/IEC 9126. Mediante el ingreso de datos de Malwares y la mejora iterativa de las técnicas de detección para la posterior eliminación, ofreciendo así un software estable y eficiente.

Finalmente se analizado, diseñado e implementado el Software Antivirus que detecta una considerable variedad de **Malwares** de acuerdo a las pruebas en un **80%**, siendo **50%** detección por firmas de virus y el **30%** por el Escanner Heurístico Proactivo, Además el actualizador permite que una versión del antivirus pueda ser actualizada era necesario crear una base de firmas, de modo que la Heurística no sufra cambios y se mejore el proceso de detección y eliminación, se ha escrito rutinas que permitieron que el antivirus pueda ejecutarse en distintos sistemas operativos y de modo autónomo para una detección en modo pasivo. la Proactividad ha permitido escribir el detector en tiempo real para las unidades de almacenamiento extraíble para prevenir infecciones en la PC local, dando la seguridad de la protección mediante las técnicas desarrolladas y descritas en el presente trabajo de investigación.

INDICE

CAPÍTULO I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1. INTRODUCCIÓN

La seguridad informática como investigación tiene como principal labor la seguridad de la información y estabilidad del sistema operativo host, evitar la infiltración de programas sospechosos o potencialmente peligrosos, todas estas labores usualmente se encargan a un experto en seguridad informática.

El usuario normal requiere un Software experto en seguridad informática, los algoritmos heurísticos y proactivos darán la inteligencia necesaria a nuestro sistema antivirus, en la primera etapa aprende, posteriormente aplica el conocimiento adquirido en la segunda etapa la de Proactividad debe adelantarse a situaciones que puedan ser peligrosas, es decir intentos de infiltración bajo archivos guiones, ejecutables y dispositivos extraíbles. Sucederán ocasiones post-infección en los que se requiere una vacuna que no dependa del sistema operativo ya infectado, para ello un antivirus provisto de su propio sistema operativo, esto le dará al producto antivirus una mayor eficiencia frente a sus competidores, sumando la capacidad de poder usarlo en diversos Sistemas Operativos.

1.2. DEFINICIÓN DEL PROBLEMA

El auge de las computadoras personales y la masificación de las mismas en la Internet traen consigo una efectiva comunicación por la Internet, la capacidad general de conseguir y compartir información, sin discriminar el género de esta. Lo que nos conlleva a plantear el objetivo de nuestra investigación. En base al cúmulo de información existente. Asimismo existen peligros en términos generales conocido como **virus informáticos, gusanos, caballos de troya, scripts maliciosos, exploits y backdoors**, los que de acuerdo a los parámetros de su programación serán capaces de alterar y/o destruir la información y/o des-configurar el sistema operativo, generando así la perdida de información, tiempo y dinero a los usuarios

La excesiva proliferación de virus informáticos en la actualidad, la masificación de las comunicaciones por Internet y el uso de dispositivos extraíbles de almacenamiento masivo han incrementado en forma rápida los ambientes para la ejecución y encasillamiento de virus y gusanos informáticos en las diversas redes de computadores, llegando a las empresas, instituciones públicas y privadas e incluso en el hogar.

Desde la última década, los virus informáticos son cada vez más complejos, difíciles de detectar y eliminar. Además cada vez proliferan más nuevos peligros, no catalogados precisamente como virus, pero también dañinos. El **spam** o correo no deseado, los **spyware** o programas espías, los **war drivers** o piratas de redes inalámbricas, los **snoops** o husmeadores, los hackers o piratas informáticos y el

phishing o correo masivo fraudulento. Se enseña cómo limpiar el ordenador, detener futuras infecciones, bloquear un sistema operativo poco seguro y controlar los daños significativos. Está concebido con un lenguaje sencillo para que aprenda a defenderse y proteger su ordenador sin necesidad de ser un experto.

Es necesario un software de bajo consumo de recursos y funcional que permita proteger la información y complementar los defectos de otras marcas antivirus, es por esto que iniciamos el desarrollo de una solución informática eficiente.

1.3. JUSTIFICACIÓN

Todos los usuarios requieren transportar continuamente diversos tipos de información mediante dispositivos de almacenamiento externo (memorias USB[1]) sin mediar el riesgo que puede acarrear el conectarla en diferentes computadoras, es necesario un software antivirus, particularmente adaptado a la zona geográfica en la que radican. En el caso particular del Perú, hay virus y gusanos que no están catalogados en la mayoría de los Scanners (Antivirus Engines) comerciales. Motivo por el cual nuestra propuesta estima catalogarlos periódicamente para una efectiva labor de limpieza en dispositivos extraíbles USB, además del disco duro mientras se trabaja de forma cotidiana. Ya que causa problemas para el usuario y la desesperación ante la pérdida de información, el antivirus permitirá recuperar la información dañada o alterada deliberadamente.

[1] USB : Universal Serial Bus se traduce como: Bus Serial Universal

El producto de la investigación colaborara en la estabilidad y seguridad de la información de computadoras personales en las facultades, centros de cómputo, oficinas de nuestra universidad y lo que es más importante aun a la información de los usuarios generales, fuera de nuestra universidad.

Desarrollar un Sistema Antivirus (Kit de Herramientas Antivirus) que pueda asegurar y proteger la información y programas de los usuarios mediante algoritmos Heurísticos y Proactivos, será una labor compleja pero de gran utilidad, por esta razón nos planteamos la interrogante:

¿Desarrollando e Implementando un Sistema Antivirus realizará de manera eficiente la detección, eliminación, prevención y actualización para las computadores personales mediante las técnicas de detección Heurística, Proactiva y la actualización por Internet de la base de firmas de Malwares?

1.4. OBJETIVO GENERAL

Desarrollar e implementar un sistema eficiente de antivirus para la detección, eliminación, prevención y actualización para las computadores personales mediante las técnicas de detección Heurística, Proactiva y la actualización por Internet de la base de firmas de Malwares.

1.5. OBJETIVOS ESPECÍFICOS

- Analizar y Diseñar el esquema más óptimo para implementar, codificar un Software Antivirus que permita proteger a los usuarios de computadores de los Malwares abundantes en los dispositivos de almacenamiento extraíbles.

- Identificar diversas técnicas de detección de virus, además de prevención de infección, se recolectará información en libros especializados en seguridad informática para Proponer un algoritmo eficaz que permita desarrollar un escáner antivirus.

- Desarrollar controladores personalizados para la detección de nuevo hardware requiere trabajar con los más altos niveles de privilegio del sistema operativo, usualmente llegando al Ring0 y Ring1[2] para la ejecución en tiempo real.

- Analizar y diseñar la implementación del modulo de actualización de firmas de virus, con un sitio web así permitir a los usuarios mantener actualizado el Software Antivirus.

1.6. HIPÓTESIS

El Sistema Antivirus que se desarrolle e implemente será eficiente en la detección, eliminación, prevención y actualización para las computadores personales mediante las técnicas de detección Heurística, Proactiva y la actualización por Internet de la base de firmas de Malwares.

[2] Ring : Niveles más altos en privilegios de ejecución de procesos

1.7. VARIABLES E INDICADORES

1.7.1 IDENTIFICACION DE VARIABLES

Variable dependiente (Y)

Variable Dependiente	Indicador
Nivel de Protección	Nivel de Protección Alcanzado

Cuadro 1.1: Variable dependiente

Variables Independientes (X_i)

Variables Independientes	Indicadores
Dimensión de Calidad	- Confiabilidad - Capacidad de Respuesta - Seguridad
Técnicas de Detección	- Escaneo Clásico - Heurísticas - Proactividad
Base Teórica	Técnico y Teórico

Cuadro 1.2: Variables independientes

1.7.2 OPERACIONALIZACION DE VARIABLES

Variables Independientes	Indicador	Criterio
Dimensiones de calidad	- Confiabilidad - Capacidad de respuesta - Seguridad	1. En total desacuerdo 2. En desacuerdo 3. Indiferente 4. De acuerdo 5. Totalmente de Acuerdo
Técnicas de Detección	- Escaneo clásico - Heurística - Proactividad	1. Malo 2. Bueno 3. Óptimo
Base teórica	Técnico teórico	
Variables Independientes	**Indicador**	**Criterio**
Nivel de Protección	Nivel de protección alcanzado	1- Malo 2- Bueno 3- Optimo

Cuadro 1.3: Operacionalización de Variables

CAPÍTULO II

FUNDAMENTOS TEÓRICOS

2.1 ANTECEDENTES DE LA INVESTIGACIÓN

- La tesis de **[hjuares,04]** manifiesta que la investigación particularmente orientada a la protección de programas ejecutables, es competitiva y en constante evolución, por lo que requiere un alto nivel de investigación, también comenta sobre las amenazas de los virus informáticos que son los que principalmente infectan y desprotegen los programas. Aborda temas como la estructura de los PE EXE de Windows, los niveles de privilegios del sistema operativos, GDT, LDT1 y temas generales de seguridad informática muy importantes para nuestro proyecto de investigación.

- En la tesis **[borgelho,01]** Es muy importante ser consciente que por más que nuestra empresa sea la más segura de ataques externos, Hackers, virus, etc. (conceptos luego tratados); la seguridad de la misma será nula si no se ha previsto como combatir un incendio. La seguridad física es uno de los aspectos más olvidados a la hora del diseño de un sistema informático. Si bien algunos de los aspectos tratados a continuación se prevén, otros, como la detección de un

atacante interno a la empresa que intenta acceder físicamente a una sala de operaciones de la misma.

En nuestra investigación buscamos mejorar estas características ampliándolas con técnicas proactivas, usando para ello técnicas inteligentes, y hasta la implementación de multi-agentes inteligentes colaborativos, pero no sobresalientes en la parte investigativa sino aplicativa.

2.2 MARCO TEÓRICO

2.2.1 VIRUS

Es un programa de cómputo, un virus informático es un malware que tiene por objeto alterar el normal funcionamiento de la computadora, sin el permiso o el conocimiento del usuario. Los virus, habitualmente, reemplazan el contenido de archivos ejecutables por otros infectados con el código de este. Los virus pueden destruir, de manera intencionada, los datos almacenados en un ordenador. Requieren de formatos ejecutables como **Win32 PE EXE, COM, SCR**, las cuales infectan para asegurar su existencia, puede apreciar su accionar en la **Figura 2.1**, muy semejante a un virus biológico **[Walker, 06]**.

2.2.2 GUSANOS

Técnicamente catalogados como **Worms**[3], actúan de forma similar a los virus en su comportamiento, la diferencia esta en la reproducción de estos, pues no infectan a otros ejecutables, sino que tiene cuerpo y se arrastran por el sistema host, adjuntándose a los correos y archivos comprimidos algunas veces. Son bastante peligrosos por ser complemente autónomos, algunos tienen técnicas polimórficas para evitar la detección de los antivirus.

2.2.3 MALWARES

Del inglés **malicious software**, también llamado badware, software malicioso o software malintencionado, es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora sin el consentimiento de su propietario. Las diversas variedades de virus aseguraran su existencia al infectar y crear un portador, el cual ejecuta el código del virus y el virus lo redirección el punto de entrada en una nueva posición al programa, evitando alertar al usuario, apreciamos este accionar en la **Figura 2.1**.

Nótese también que el comportamiento se relaciona con un virus biológico, pues al no poder existir y mantenerse por si solo, infecta otro cuerpo y convive en secreto en ella, al existir contacto con otros cuerpos estos también son infectados **[Walker, 06].**

[3] **Worm :** del término en Ingles traducido como gusano.

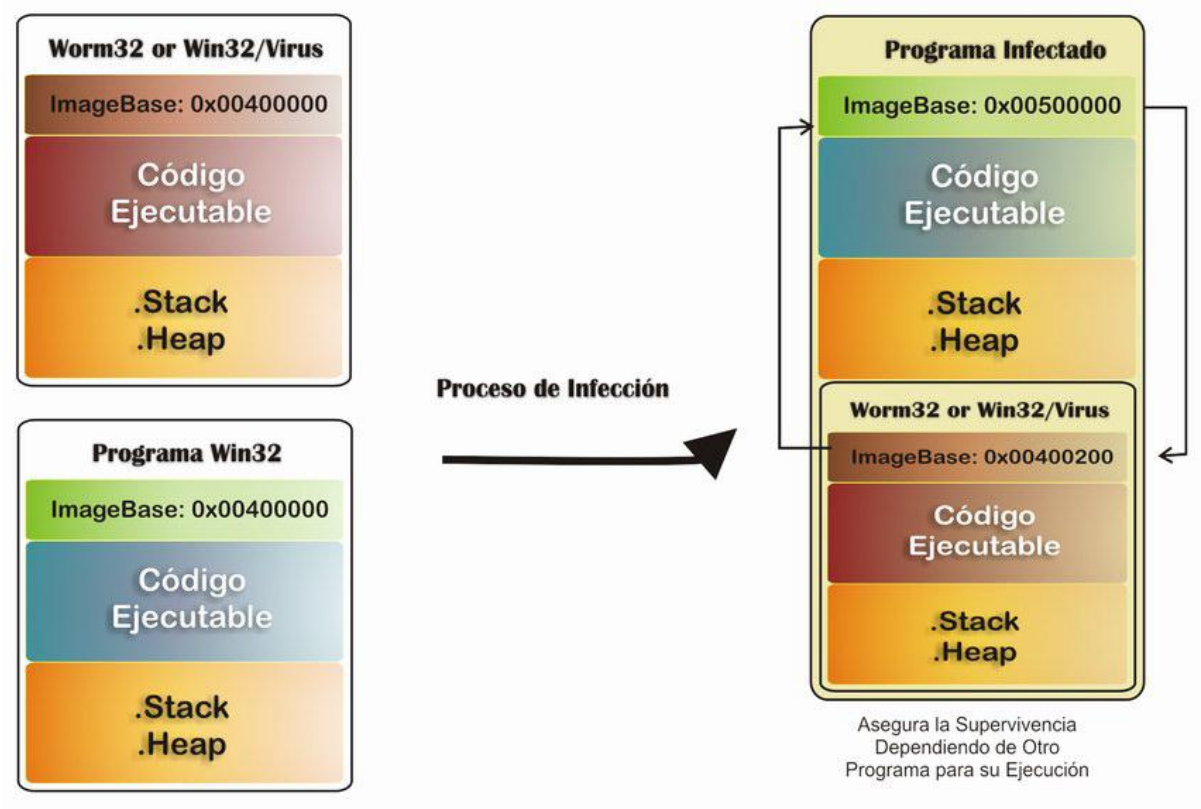


Figura 2.1: Proceso grafico de infección de un Win32/Virus

2.2.4 TIEMPO REAL

Un sistema en tiempo real (STR) es aquel sistema digital que interactúa activamente con un entorno con dinámica conocida en relación con sus entradas, salidas y restricciones temporales, para darle un correcto funcionamiento de acuerdo con los conceptos de predictibilidad, estabilidad y controlabilidad. La principal característica que distingue a los STR de otros tipos de sistemas es el tiempo de interacción. Sin embargo, antes de continuar es necesario aclarar el significado de las palabras *''**tiempo**''* y *''**real**''*. La palabra *''**tiempo**''* significa que el correcto funcionamiento de un sistema depende no sólo del resultado lógico que devuelve la computadora, también

depende del tiempo en que se produce ese resultado. La palabra ***''real''*** quiere decir que la reacción de un sistema a eventos externos debe ocurrir durante su evolución. Como una consecuencia, el tiempo del sistema (tiempo interno) debe ser medido usando la misma escala con que se mide el tiempo del ambiente controlado (tiempo externo) **[Rogers, 03]**.

2.2.5 HEURÍSTICA

Heurística es la capacidad de un sistema para realizar de forma inmediata innovaciones positivas para sus fines. La capacidad heurística es un rasgo característico de los humanos, desde cuyo punto de vista puede describirse como el arte y la ciencia del descubrimiento y de la invención o de resolver problemas mediante la creatividad y el pensamiento lateral o pensamiento divergente.

La palabra heurística significa «hallar, inventar» **(etimología que comparte con *eureka*)**. La palabra Heurística[4] aparece en más de una categoría gramatical. Cuando se usa como sustantivo, identifica el arte o la ciencia del descubrimiento, una disciplina susceptible de ser investigada formalmente. Cuando aparece como adjetivo, se refiere a cosas más concretas, como estrategias heurísticas, reglas heurísticas o silogismos y conclusiones heurísticas. Claro está que estos dos usos están íntimamente relacionados ya

[4] Heurística : Según la Real Academia,(consultado el 29 de abril de 2009)

que la heurística usualmente propone estrategias heurísticas que guían el descubrimiento.

La heurística aplicada como método inteligente en la toma de decisiones poco difundida en inteligencia artificial, pero con una adecuada implementación se convierte en una útil técnica para la toma de decisiones, en el caso particular de los antivirus, toma de una base de conocimientos para generar nuevas decisiones escalables en lo posible en la línea de tiempo **[Rogers,03]**.

2.2.6 PROACTIVIDAD

Proactividad es un término acuñado por Viktor Frankl, un neurólogo y psiquiatra austriaco que sobrevivió a los campos de concentración nazis, en su libro Man's Search for Meaning (El hombre en busca de sentido, 1946). Años después el término se popularizaría en muchos libros de autoayuda, desarrollo personal y empresarial gracias al best-seller *(éxito en ventas)* Los siete hábitos de las personas altamente efectivas del autor Stephen R. Covey.

Proactividad es una actitud en la que el sujeto asume el pleno control de su conducta vital de modo activo, lo que implica la toma de iniciativa en el desarrollo de acciones creativas y audaces para generar mejoras, haciendo prevalecer la libertad de elección sobre las circunstancias de la vida.

La Proactividad no significa sólo tomar la iniciativa, sino asumir la responsabilidad de hacer que las cosas sucedan; decidir en cada momento lo que queremos hacer y cómo lo vamos a hacer **[Rogers, 03]**.

2.2.7 SISTEMA MULTIPLATAFORMA

A) **EL ESTÁNDAR DE VIDEO VESA**

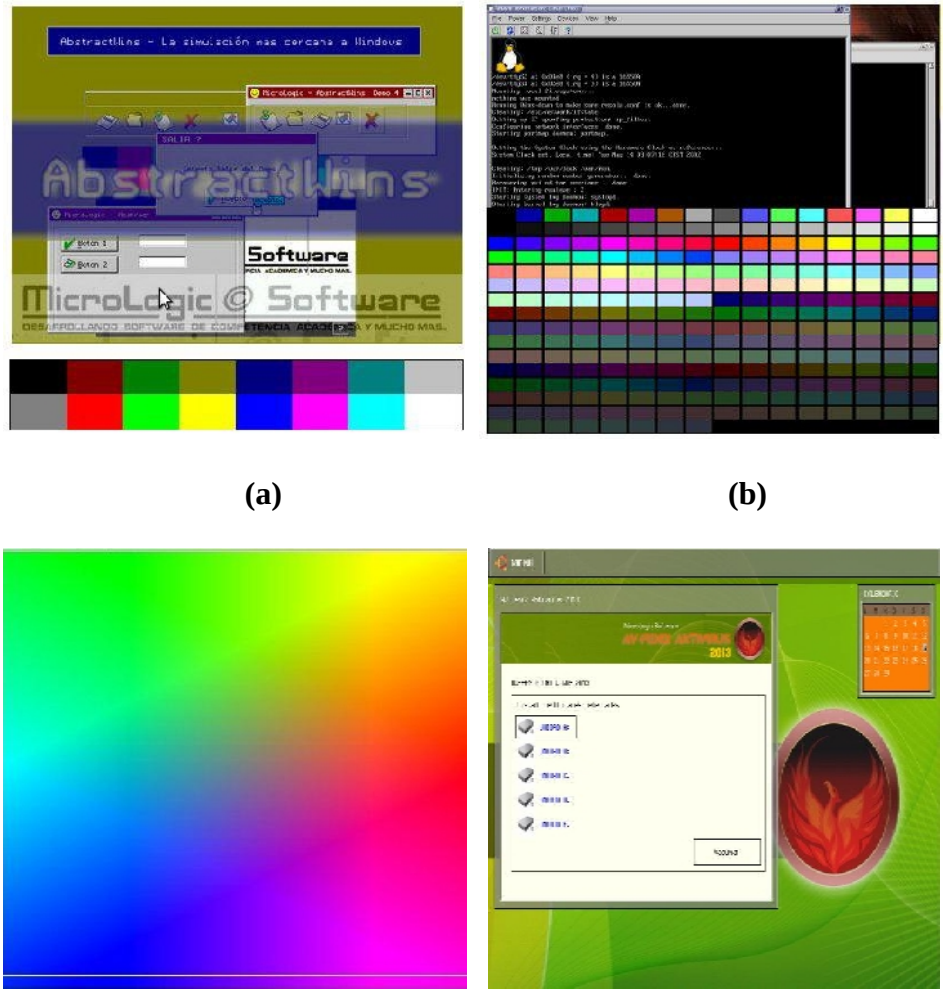
Debido a la anarquía reinante en el mundo de las tarjetas gráficas, en 1989 se reunieron un grupo importante de fabricantes (ATI, Genoa, Intel, Paradise, etc) para intentar crear una norma común. El resultado de la misma fue el estándar VESA. Este estándar define una interface software común a todas las BIOS para permitir a los programadores adaptarse con facilidad a las diversas tarjetas sin tener en cuenta sus diferencias de hardware.

Actualmente, las principales tarjetas soportan la norma VESA. Las más antiguas pueden también soportarla gracias a pequeños programas residentes que el usuario puede instalar opcionalmente. Para desarrollar una aplicación profesional, es una buena norma soportar algún modo estándar de la VGA y, para obtener más prestaciones, algún modo VESA para los usuarios que estén equipados con dicho soporte. Intentar acceder directamente al hardware o a las funciones BIOS propias de cada tarjeta del mercado por separado, salvo para aplicaciones muy concretas, es ciertamente poco menos que imposible. **[García, 02]**

B) **MODOS GRÁFICOS**

El estándar VESA soporta multitud de modos gráficos, numerados a partir de 100h, si bien algunos de los más avanzados (con 32000 o 16 millones de colores) sólo están soportados por las versiones más recientes de la norma VESA actualmente en la version 3.0 conocida como VBE 3.0 (Vesa Bios Specification). Entre 100h y 107h se definen los modos más comunes de 16 y 256 colores de todas las SuperVGA, aunque el modo 6Ah también es VESA (800x600x16) al estar soportado por múltiples tarjetas.

Una de las grandes ventajas del estándar VESA es la enorme información que pone a disposición del programador. ***(De esta investigación existe documentación VESA en el sitio filicc.webcindario.com sección C/C++)*** Es posible conocer todas las características de resolución, colores y arquitectura. Además, hay funciones adicionales muy útiles para guardar y recuperar el estado de la tarjeta, de especial utilidad para programas residentes: así, estos pueden fácilmente conmutar a modo texto (con la precaución de preservar antes los 4 primeros KBytes *(Kilobytes o equivalente a 1024 bytes)* de la RAM *(Memoria de acceso aleatorio, de escritura y/o lectura sin importar el orden de acceso)* de vídeo empleados para definir los caracteres) y volver al modo gráfico original dejando la pantalla en el estado inicial.



(a) **(b)**

(c) **(d)**

(a) Modo VGA con una paleta de 16 colores y resolución de 640x480 **(b)** Modo SVGA de 256 Colores y resolución de 1024x768. **(c)** Modo VESA paleta de 24 millones de colores. **(d)** Demo realizado con KronosVESA y KrystalView librería con resoluciones variantes de 640x480 hasta 2048x1024

C) **INTERFAZ GRAFICA DE USUARIO (GUI)** :

La interfaz gráfica de usuario, conocida también como GUI (del inglés graphical user interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su

principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador. Habitualmente las acciones se realizan mediante manipulación directa, para facilitar la interacción del usuario con la computadora. Surge como evolución de las interfaces de línea de comandos que se usaban para operar los primeros sistemas operativos y es pieza fundamental en un entorno gráfico. Como ejemplos de interfaz gráfica de usuario, cabe citar los entornos de escritorio Windows, el X-Window de GNU/Linux o el de Mac OS X, Aqua. **[tanembaum, 06]**

(a)

(b)

(c)

(d)

(a) Xerox Primer GUI 1980 **(b)** Mac OS Lion 2005. **(c)** Windows Metro 2012. **(d)** GNOME Desktop GNU/Linux 2012

D) **DISCO VIRTUAL O RAMDRIVE**

Un disco RAM o unidad RAM es un bloque de memoria RAM (almacenamiento primario o memoria volátil) que el software de una computadora está tratando como si la memoria fuera una unidad de disco (almacenamiento secundario). Se refiere a veces como un disco RAM virtual o software de la unidad de RAM para distinguirlo de un **hardware de la unidad de RAM** que utiliza el hardware por separado que contiene memoria RAM, que es un tipo de unidad de estado sólido pero volátil al reiniciar la PC **[Rodríguez, 88]**.

El rendimiento de un disco RAM se encuentra en las órdenes generales de magnitud más rápido que otras formas de medios de almacenamiento, tales como un disco SSD, disco duro, unidad de cinta o una unidad óptica. Este aumento del rendimiento se debe a múltiples factores, incluyendo el tiempo de acceso, rendimiento máximo y el tipo de sistema de fichero, así como otros. El tiempo de acceso a archivos está muy disminuido desde un disco RAM es de estado sólido (sin partes mecánicas). Un disco duro físico o en medios ópticos, como CD-ROM, DVD y Blu-ray debe mover la cabeza o los ojos óptica en la posición y unidades de cinta debe terminar o retroceder a una posición particular sobre los medios de comunicación antes de la lectura o escritura puede ocurrir. Discos RAM se puede acceder a los datos con sólo la dirección de memoria de un archivo determinado, sin movimiento, la alineación o posicionamiento necesario **[Abel, 02]**.

2.2.8 DATOS E INFORMACIÓN

Un dato puede considerarse como un elemento no tratado; como una señal emitida. Para que un dato tenga algún tipo de validez, tiene que presentarse en un contexto. Debe ser procesado y analizado para poder inferir una conclusión del mismo. En el momento en que un dato es procesado, se le puede considerar información. Su importancia está en la capacidad de asociarse dentro de un contexto para convertirse en información.

La información es un mensaje con un contenido determinado *(usualmente es un conjunto de octetos*[5]*)* emitido por una persona hacia otra y, como tal representa un papel primordial en los procesos de la comunicación, a la vez que posee una evidente función social. A diferencia de los datos, la información tiene información para quien lo recibe, por eso, los seres humanos siempre han tenido la necesidad de cambiar entre si información que luego transforman en acciones.

"La información es, entonces, conocimientos basados en los datos a los cuales, mediante un procesamiento, se les ha dado significado, propósito y utilidad."

[5] Octeto : es la traducción de la palabra Inglesa **Byte**, conjunto de 8 estados binarios entre 0 y 1

2.2.9 DAÑOS OCASIONADOS POR LOS VIRUS

Los virus informáticos no afectan directamente el hardware sino a través de los programas que lo controlan; en ocasiones no contienen código nocivo, o bien, únicamente causan daño al reproducirse y utilizar recursos escasos como el espacio en el disco rígido, tiempo de procesamiento, memoria, etc. En general los daños que pueden causar los virus se refieren a hacer que el sistema se detenga, borrado de archivos, comportamiento erróneo de la pantalla, despliegue de mensajes, desorden en los datos del disco, aumento del tamaño de los archivos ejecutables **[Borgelho, 01]**.

Para realizar la siguiente clasificación se ha tenido en cuenta que el daño es una acción de la computadora, no deseada por el usuario:

A) DAÑO IMPLÍCITO:

Es el conjunto de todas las acciones dañinas para el sistema que el virus realiza para asegurar su accionar y propagación. Aquí se debe considerar el entorno en el que se desenvuelve el virus ya que el consumo de ciclos de reloj en un medio delicado (como un aparato biomédico) puede causar un gran daño.

B) DAÑO EXPLÍCITO:

Es el que produce la rutina de daño del virus.

Con respecto al modo y cantidad de daño, encontramos:

A) DAÑOS TRIVIALES:

Daños que no ocasionan ninguna pérdida grave de funcionalidad del sistema y que originan una pequeña molestia al usuario. Deshacerse del virus implica, generalmente, muy poco tiempo.

B) DAÑOS MENORES:

Daños que ocasionan una pérdida de la funcionalidad de las aplicaciones que poseemos. En el peor de los casos se tendrá que reinstalar las aplicaciones afectadas.

C) DAÑOS MODERADOS:

Los daños que el virus provoca son formatear el disco rígido o sobrescribir parte del mismo. Para solucionar esto se deberá utilizar la última copia de seguridad que se ha hecho y reinstalar el sistema operativo.

D) DAÑOS MAYORES:

Algunos virus pueden, dada su alta velocidad de infección y su alta capacidad de pasar desapercibidos, lograr que el día que se detecta su presencia tener las copias de seguridad también infectadas. Puede que se llegue a encontrar una copia de seguridad no infectada, pero será tan antigua que se haya perdido una gran cantidad de archivos que fueron creados con posterioridad.

E) DAÑOS SEVEROS:

Los daños severos son hechos cuando un virus realiza cambios mínimos, graduales y progresivos. No se sabe cuando los datos son correctos o han cambiado, pues no hay unos indicios claros de cuando se ha infectado el sistema.

2.2.10 ARQUITECTURA CLIENTE/SERVIDOR

Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

Su capacidad de procesamiento está repartida entre los clientes y servidores, es debido a estos últimos que posee la propiedad de centralizar la gestión de la información.

La red cliente-servidor es aquella red de comunicaciones en la que todos los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados.

Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta.

A) **MODELO SIMPLE DE INTERACCIÓN CLIENTE/SERVIDOR**

"Una aplicación cliente servidor se basa en el modelo de solicitud – respuesta, el caso más simple corresponde a la situación en la cual una aplicación (el cliente) solicita un recurso y otra (el servidor) la atiende para brindarle el servicio de ser posible".

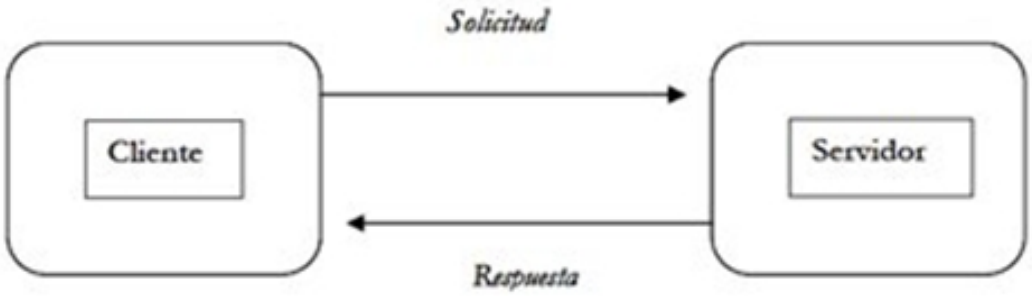


Figura 2.4: Esquema de comunicación de la arquitectura Cliente/Servidor

B) **PROPIEDADES DE LOS SISTEMAS CLIENTE/SERVIDOR**

- Transparencia de ubicación, hardware y plataforma (SO).
- Una petición hecha por un cliente o servidor indica claramente qué servicio se desea y entonces un servidor se encarga de cómo resolverla.
- Se pueden realizar acciones de escalamiento (upgrade) sin afectar a otros componentes, se manejan dos tipos:

  - **Horizontal:** se agregan otros clientes y servidores.
  - **Vertical:** se cambia un servidor por otro más potente o se distribuye su trabajo entre varios.

- Las funciones y datos del servidor son manejadas en forma centralizada. Eso beneficia el mantenimiento e integridad de los datos.

C) **VENTAJAS DEL ESQUEMA CLIENTE/SERVIDOR**

a) El uso de los sistemas cliente servidor fue masificado debido a la producción de hardware cada vez más económico. Esto constituye a su vez una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas grandes. Además, se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.

b) La arquitectura cliente servidor ofrece la facilidad de integrarse y operar entre diferentes plataformas. De esta manera, podemos integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operacional.

c) Interfaces gráficas interactivas e intuitivas para el usuario final, las cuales no necesariamente son enviadas a través de la red y pueden residir en el cliente, permitiendo aprovechar el ancho de banda.

d) Una ventaja adicional del uso del esquema Cliente/Servidor es que es más rápido el mantenimiento y el desarrollo de aplicaciones, haciendo uso de herramientas existentes como los servidores SQL.

2.2.11 INTERNET

Internet es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial. Sus orígenes se remontan a 1969, cuando se estableció la primera conexión de computadoras, conocida como ARPANET, entre tres universidades en California y una en Utah, Estados Unidos.

Uno de los servicios que más éxito ha tenido en Internet ha sido la World Wide Web (WWW, o "la Web" desarrollado por Tim Berners--Lee en 1989, Científico Informático en el Laboratorio Europeo de Física de Partículas **CERN**[6]), hasta tal punto que es habitual la confusión entre ambos términos. La WWW es un conjunto de protocolos que permite, de forma sencilla, la consulta remota de archivos de hipertexto. Ésta fue un desarrollo posterior (1990) y utiliza Internet como medio de transmisión.

Existen, por tanto, muchos otros servicios y protocolos en Internet, aparte de la Web: el envío de correo electrónico (SMTP), la transmisión de archivos (FTP y P2P), las conversaciones en línea (IRC), la mensajería instantánea y presencia, la transmisión de contenido y comunicación multimedia -telefonía (VoIP), televisión (IPTV)-, los boletines electrónicos (NNTP), el acceso remoto a otros dispositivos (SSH y Telnet) o los juegos en línea.

Los métodos comunes de acceso a Internet en los hogares incluyen dial-up, banda ancha fija (a través de cable coaxial, cables de fibra óptica o cobre), **Wi-Fi** televisión vía satélite y teléfonos celulares con tecnología 3G/4G. Los lugares públicos de uso del Internet incluyen bibliotecas y cafés de internet, donde los ordenadores con conexión a Internet están disponibles.

[6] **CERN** : de su nombre de origen Francés: **Centre Eorupéenne pour Recherche Nucleairé** (Centro Europeo para Investigación Nuclear) Laboratorio científico ubicado en Suiza, aloja el supe acelerador de partículas más poderoso construido por la Humanidad.

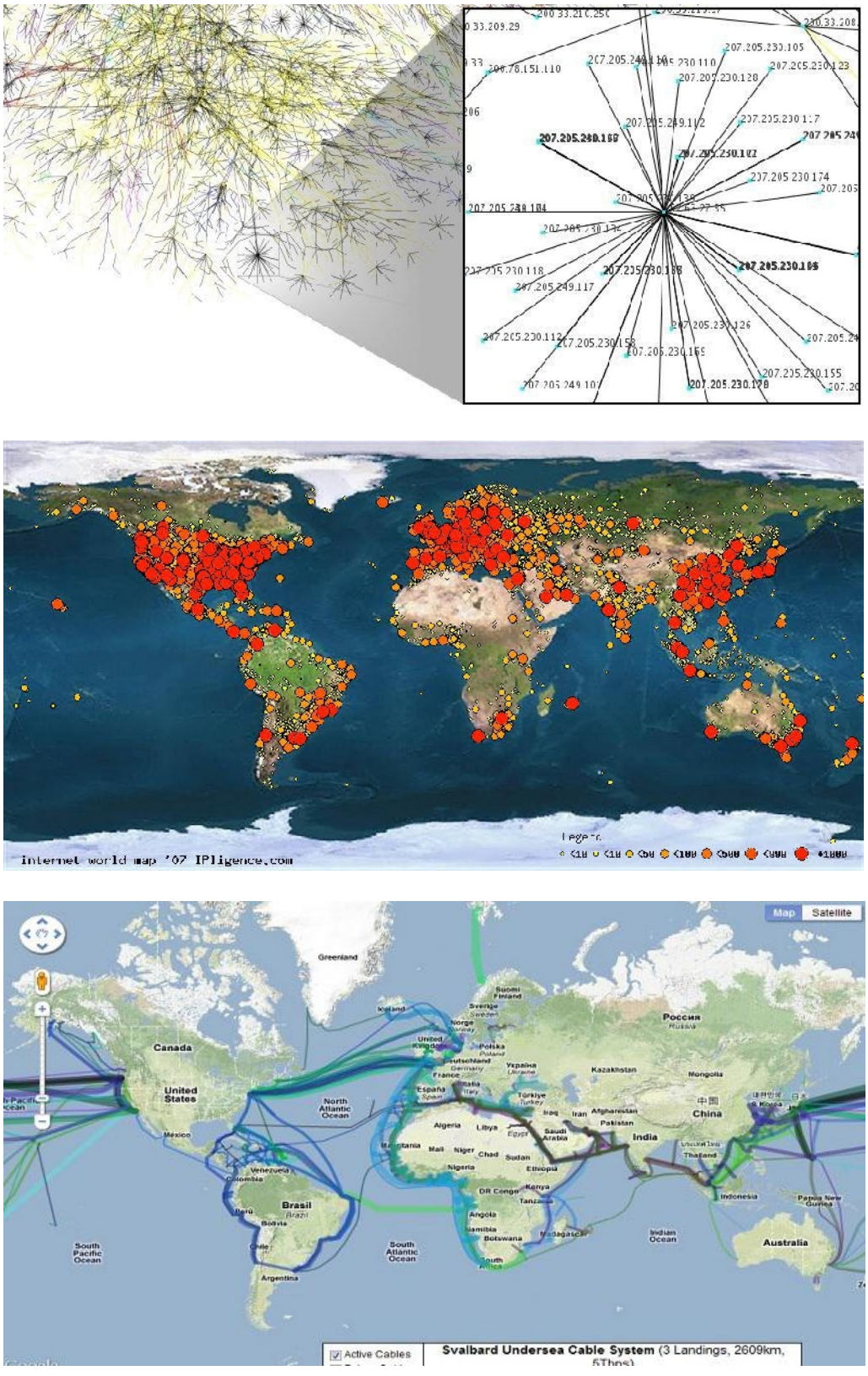


Figura 2.5: Internet visión grafica y mundial de su alcance.

2.2.12 SOCKETS

Los sockets no son más que puntos o mecanismos de comunicación entre procesos que permiten que un proceso hable *(emita o reciba información)* con otro proceso incluso estando estos procesos en distintas máquinas. Esta característica de interconectividad entre máquinas hace que el concepto de socket nos sirva de gran utilidad. Esta interfaz de comunicaciones es una de las distribuciones de Berkeley al sistema UNIX, implementándose las utilidades de interconectividad de este Sistema Operativo (rlogin, http, telnet, ftp) usando sockets **[Jsmart, 06]**.

Un socket es al sistema de comunicación entre ordenadores lo que un buzón o un teléfono es al sistema de comunicación entre personas: un punto de comunicación entre dos agentes *(procesos o personas respectivamente)* por el cual se puede emitir o recibir información. La forma de referenciar un socket por los procesos implicados es mediante un descriptor del mismo tipo que el utilizado para referenciar ficheros. Debido a esta característica, se podrá realizar redirecciones de los archivos de E/S estándar (descriptores 0, 1 y 2) a los sockets y así combinar entre ellos aplicaciones de la red. Todo nuevo proceso creado heredará, por tanto, los descriptores de sockets padre heredaran los métodos y la conexión una vez que ***Accept*** haya sido activada, permitiendo con el ***Send*** el envió y compartición de información **[Jsmart, 06]**.

2.2.13 EL PROTOCOLO TCP/UDP

TCP/IP ó Protocolo de Control de Conexión ***(User Control Protocol)***, TCP da soporte a muchas de las aplicaciones más populares de Internet (navegadores, intercambio de ficheros, clientes ftp) y protocolos de aplicación HTTP, SMTP, SSH y FTP.

Es un protocolo de comunicación orientado a conexión y fiable del nivel de transporte, actualmente documentado por IETF, que requiere de la validación de integra entrega de los paquetes enviados. Es un protocolo de capa 4 según el modelo OSI **[Burns, 03]**, mostrado en la Figura 2.6.

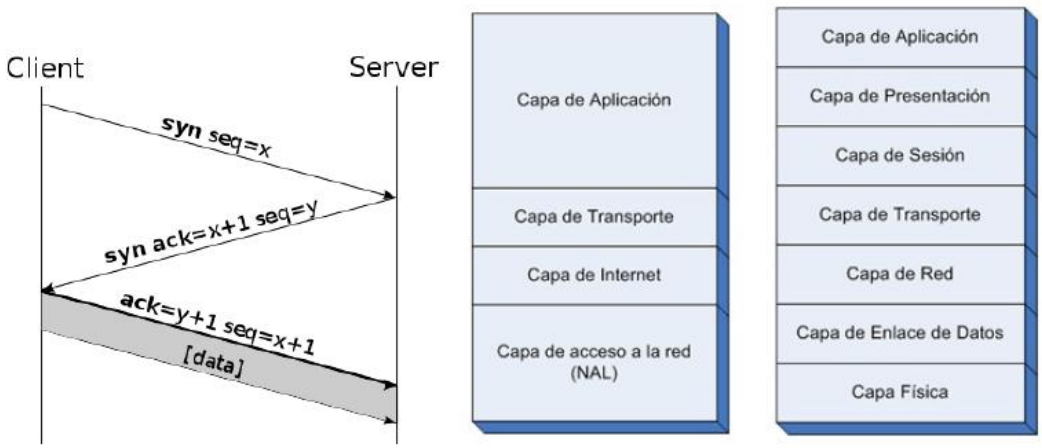


Figura 2.6: Esquema general de protocolo TCP/IP

A) FUNCIONES DE TCP

En la pila de protocolos TCP/IP, TCP es la capa intermedia entre el protocolo de internet (IP) y la aplicación. Habitualmente, las aplicaciones necesitan que la comunicación sea fiable y, dado que la capa IP aporta un servicio de datagramas no fiable (sin confirmación), TCP añade las funciones necesarias para prestar un servicio que permita que la comunicación entre dos sistemas se efectúe libre de errores, sin pérdidas y con seguridad.

Orientado a la conexión: dos computadoras establecen una conexión para intercambiar datos. Los sistemas de los extremos se sincronizan con el otro para manejar el flujo de paquetes y adaptarse a la congestión de la red.

- **Operación Full-Dúplex :** una conexión TCP es un par de circuitos virtuales, cada uno en una dirección. Sólo los dos sistemas finales sincronizados pueden usar la conexión.

- **Error Checking :** una técnica de **checksum** es usada para verificar que los paquetes no estén corruptos.

- **Acknowledgements :** sobre recibo de uno o más paquetes, el receptor regresa un acknowledgement **(reconocimiento)** al transmisor indicando que recibió los paquetes. Si los paquetes no son

notificados, el transmisor puede reenviar los paquetes o terminar la conexión si el transmisor cree que el receptor no está más en la conexión.

- **Control de flujo :** si el transmisor está desbordando el buffer del receptor por transmitir demasiado rápido, el receptor descarta paquetes. Los **acknowledgement** fallidos que llegan al transmisor le alertan para bajar la tasa de transferencia o dejar de transmitir.

- **Servicio de recuperación de Paquetes :** el receptor puede pedir la retransmisión de un paquete. Si el paquete no es notificado como recibido (ACK), el transmisor envía de nuevo el paquete. Los servicios confiables de entrega de datos son críticos para aplicaciones tales como transferencias de archivos (FTP por ejemplo), servicios de bases de datos, proceso de transacciones y otras aplicaciones de misión crítica en las cuales la entrega de cada paquete debe ser garantizada.

UDP/IP ó Protocolo de Datagrama de Usuario ***(User Datagram Protocol)*** es un protocolo del nivel de transporte basado en el intercambio de datagramas (Paquete de datos). Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en

su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción. Su uso principal es para protocolos como DHCP, BOOTP, DNS, Video Streaming y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo de los paquetes en sus diversas etapas por la red, como se detalla en **[keneth, 01]**.

2.2.14 wxWidgets Cross-Platform

Es una clase wxWidgets basada en sockets BSD (Berkley Sockets) de Unix adaptado una clase que soporta una conexión multi-hilo, evitando el bloqueo asíncrono del programa, la transcripción de la especificación **wxSocket** puede encontrarse en el texto **[Jsmart, 06 : pag. 463]** y dice:

*At the core of socket operations is **wxSocketBase**, which provides the basic socket functionality for sending and receiving data, closing, error reporting, and so on. Establishing a listening socket or connecting to a server requires wxSocketServer or wxSocketClient, respectively. wxSocketEvent is used to notify the application of an event that has occurred on a socket. The abstract class **wxSocketBase** and its children such as wxIPV4address enable you to*

specify remote hosts and ports. Lastly, stream classes such as wxSocketInputStream and wxSocketOutputStream can be coupled with other streams to move and transform data over a socket. Streams were discussed in Chapter 14, Files and Streams.

Sockets in wxWidgets can operate in different ways, as discussed later in the "Socket Flags" section. The traditional threaded socket approach is handled by disabling the socket events and using blocking socket calls. On the other hand, you can enable socket events and eliminate the need for a separate thread; wxWidgets will send an event to your application when processing is required on a socket. By letting the data arrive in the background and processing data only when it is present, you avoid blocking the GUI, and you avoid the complexity of putting each socket in its own thread.

Puede traducirse como **wxSocketBase** es la clase nativa para los socket en modo cliente o servidor, es la que encapsula las clases que soportan la conección y la suspensión de la espera en un hilo separado de la aplicación. Además detalla que es aún mejor que los sockets de M.F.C. *(Microsoft Foundation Class librería de clases para aplicaciones Win32 desarrollada por Microsoft)* por soportar **wxSocketStreams** para poder transportar grandes cantidades de información sin tener problemas entre los distintos sistemas operativos.

2.2.15 RING

En informática, los dominios jerárquicos de niveles de privilegios (un vistazo general en la Figura 2.7), a menudo se llama anillos de protección, son un mecanismo para proteger los datos y la funcionalidad de las fallas (tolerancia a fallos) y el comportamiento malicioso (seguridad informática). Este enfoque es diametralmente opuesta a la de la capacidad de la seguridad basada en:

Los sistemas operativos proporcionan diferentes niveles de acceso a los recursos. Un anillo de protección es una de dos o más niveles jerárquicos o capas de privilegio dentro de la arquitectura de un sistema informático. Esto generalmente es forzado por hardware por parte de algunos CPU arquitecturas que ofrecen diferentes modos de la CPU en el hardware o el microcódigo nivel. Los anillos están dispuestos en una jerarquía de la mayoría de los privilegiados (de mayor confianza, por lo general número cero) por lo menos a información privilegiada (por lo menos de confianza, por lo general con el mayor número de anillo). En la mayoría de sistemas operativos, Ring 0 es el nivel con la mayoría de los privilegios e interactúa más directamente con el hardware físico, tales como la CPU y la memoria.

Puertas especiales entre los anillos se proporcionan para permitir un anillo exterior para acceder a recursos de un anillo interior de una manera predefinida, en lugar de permitir el uso arbitrario. Correctamente el acceso **Gating** (compuertas) entre los anillos puede mejorar la seguridad mediante la

prevención de los programas de un anillo o nivel de privilegios de uso indebido de los recursos destinados a los programas en sí. Por ejemplo, el spyware se ejecuta como un programa de usuario en el anillo 3 se debe impedir el giro de una cámara web sin informar al usuario, ya que el acceso de hardware debería ser un anillo de una función reservada a los controladores de dispositivos . Programas como navegadores web que se ejecutan en mayor número los anillos deben solicitar el acceso a la red, un recurso restringido a un anillo de menor número.

Todos los conceptos aquí vertidos están detallados en **[Rogers, 03]**, puede buscarse la versión electrónica para mayor información.

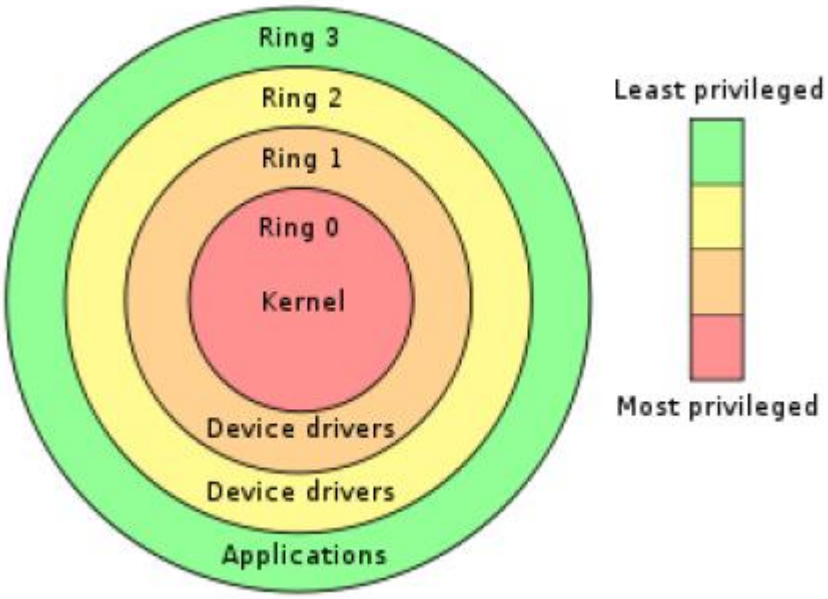


Figura 2.7: Anillo de Privilegios para el Intel x86 disponible en modo Protegido

2.2.16 BOOTSTRAP

La palabra inglesa bootstrapping es generalmente un término utilizado para describir el arranque, o proceso de inicio de cualquier ordenador. Suele referirse al programa que arranca un sistema operativo como por ejemplo GRUB, LiLo o NTLDR. Se ejecuta tras el proceso POST del BIOS. También es llamado "Bootstrap Loader" (cargador de inicialización).

Una vez que el PC arranca, comienza a ejecutarse el código que se encuentra en la dirección F000:FFF0, que pertenece al ROM-BIOS y es el encargado de realizar una serie de pruebas e inicializaciones. Esta rutina se llama POST (Power On Self-Test). Una vez que el BIOS termina con sus tests e inicializaciones carga el primer sector (cilindro 0, cabeza 0, sector 1) en el segment y offset. **0000:7C00** ***(7C00 lineal)***, comprueba que contenga código válido (comprueba que esté firmado con 0x55 y 0xAA en los bytes 511 y 512 respectivamente) y salta a esa dirección (CS:IP apuntan a esa dirección).

2.2.17 SECTOR DE ARRANQUE

Un sector es un espacio físico de 512 bytes, Un Master Boot Record *(MBR)*, es el primer sector "sector cero", de un dispositivo de almacenamiento de datos, como un disco duro. A veces es empleado para el arranque del sistema operativo con Bootstrap, otras veces es usado para guardar una tabla de particiones y en ocaciones, se usa solo para identificar un dispositivo de disco individual, aunque en algunas máquinas.

Este sector es especial pues contiene un código de arranque que es un programa de 512 bytes como máximo, por lo general (pero no necesariamente), de un sistema operativo almacenado en otros sectores del disco. El término sector de arranque es usado para los Compatible IBM PC, mientras que bloque de arranque es usado cuando se refiere a otros tipos de computadoras, como los sistemas de Sun Microsystems.

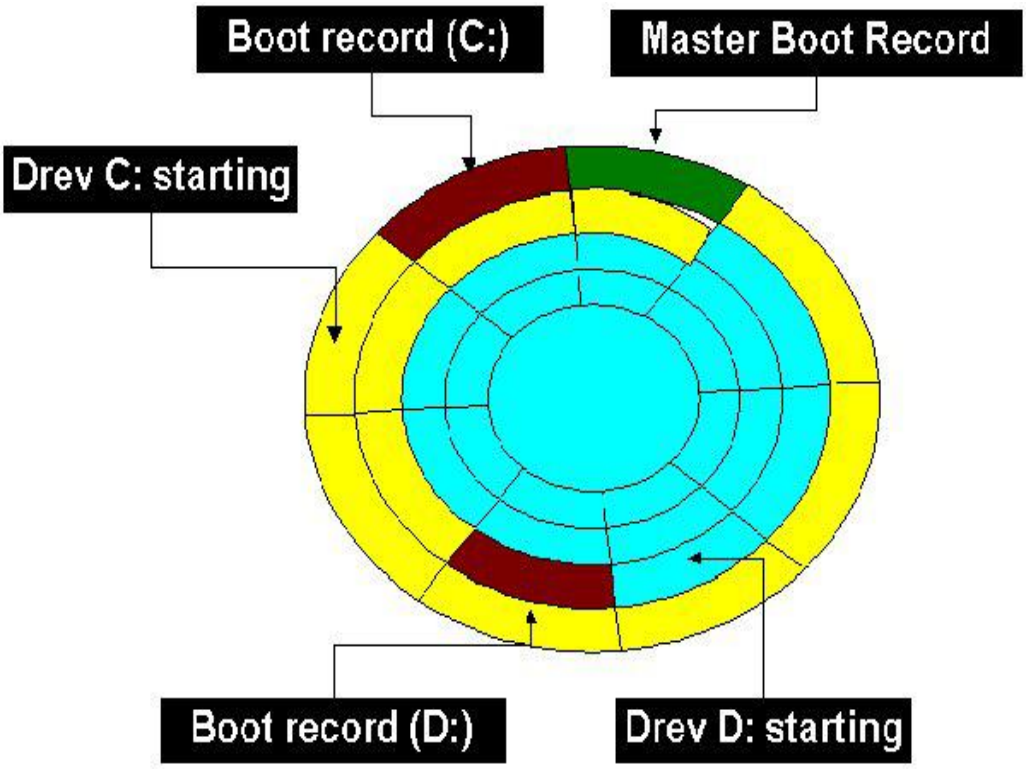


Figura 2.8: Distribución del MBR Master Boot Record

Un registro de arranque principal, conocido también como registro de arranque maestro o por su nombre en inglés es el primer sector ("sector cero") de un dispositivo de almacenamiento de datos, como un disco duro, se emplea para el arranque del sistema operativo con **bootstrap**, otras veces es usado para almacenar una tabla de particiones.

2.2.18 SECUENCIA DE ARRANQUE

Cuando arrancamos el ordenador, éste busca en las unidades de almacenamiento los ficheros necesarios para poner en marcha el sistema, según una secuencia de arranque fijada por la BIOS. Normalmente es A:> C:, para que primero lea la unidad de disquetes, por si queremos formatear el disco duro o instalar un sistema operativo. Posteriormente pasa a la unidad C: que es donde está instalado el sistema operativo, por esa razón no debe de dejarse ningún disquete dentro de la disquetera, ya que si el sistema lo detecta intenta arrancar desde ese disco, y no cargara Windows. La BIOS selecciona un dispositivo de arranque, entonces copia al primer sector de disco desde el dispositivo (el cual puede ser un **MBR** o un código ejecutable), a la ubicación de dirección de disco **0000:7C00**, el cual contiene un programa como:

Ejemplo 1 (Programa fuente de un bootstrap).

```
; nombre de programa : boot.asm compilar en nasm

   mov  ax, 0x07C0  ; sector:offset
   mov  ds, ax      ; set data segment

   mov  si, msg     ; segment index to msg
jnloop:
   lodsb            ; load string bytes
   or   al, al      ; zero=end or str
   jz   hang        ; get out
   mov  ah, 0x0E    ; print TTY
   int  0x10        ; video int
   jmp  jnloop

hang:
   jmp  hang

   msg    db  'We are Starting AvFenix...', 0
   times  510-($-$$) db 0  ; diference 510 - code
   db     0x55, 0xAA       ; boot signature
```

2.2.19 MÉTRICA DE SOFTWARE ISO/IEC 9126

ISO 9126 es un estándar internacional para la evaluación de la calidad del software. Está reemplazado por el proyecto SQuaRE, ISO 25000:2005, el cual sigue los mismos conceptos.

El estándar está dividido en cuatro partes las cuales dirigen, respectivamente, lo siguiente: modelo de calidad, métricas externas, métricas internas y calidad en las métricas de uso y expendido.

El modelo de calidad establecido en la primera parte del estándar, ISO 9126, clasifica la calidad del software en un conjunto estructurado de características y sub-características de la siguiente manera:

A) **Funcionalidad:** Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen las necesidades implícitas o explícitas.

- Idoneidad
- Exactitud
- Interoperabilidad
- Seguridad
- Cumplimiento de normas

B) **Fiabilidad:** Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.

- Madurez
- Recuperabilidad
- Tolerancia a fallos

C) **Usabilidad :** Un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios.

- Aprendizaje
- Comprensión
- Operatividad
- Atractividad

D) **Eficiencia :** Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

- Comportamiento en el tiempo
- Comportamiento de recursos
- Tolerancia a fallos

E) **Mantenibilidad :** Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

- Estabilidad
- Facilidad de análisis
- Facilidad de cambio
- Facilidad de pruebas

F) **Portabilidad:** Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.

- Capacidad de instalación
- Capacidad de reemplazamiento
- Adaptabilidad
- Facilidad de pruebas
- Co-Existencia

Este estándar proviene desde el modelo establecido en 1977 por McCall y sus colegas, los cuales propusieron un modelo para especificar la calidad del software. El modelo de calidad McCall está organizado sobre tres tipos de Características de Calidad. ISO 9126 distingue entre fallo y no conformidad. Un fallo es el incumplimiento de los requisitos previos, mientras que la no conformidad es el incumplimiento de los requisitos especificados. Una distinción similar es la que se establece entre validación y verificación.

2.2.20 INVESTIGACIÓN

La investigación versa sobre un objeto reconocible y definido de tal modo que también sea reconocible por los demás. El término objeto no tiene necesariamente un significado físico. También la raíz cuadrada es un objeto aunque nadie la haya visto nunca. La clase social es un objeto de investigación, aunque alguno pudiera objetar que sólo se conocen individuos o medias estadísticas y no clases en sentido estricto. Pero según esto tampoco tendría realidad física la clase de todos los números enteros superiores al 3725, de la cual, sin embargo, un matemático se podría ocupar estupendamente. Definir el objeto significa entonces definir las condiciones bajo las cuales podemos hablar en base a unas reglas que nosotros mismos estableceremos o que otros han establecido antes que nosotros. Si establecemos las reglas en base a las cuales un número entero superior al 3725 puede ser reconocido cuando se encuentra, hemos establecido las reglas de reconocimiento de nuestro objeto **[Eco, 05]**

Naturalmente surgen problemas si tenemos que hablar, por ejemplo, de un ser fabuloso cuya inexistencia reconoce la opinión común, como por ejemplo el centauro. Llegados a este punto tenemos tres alternativas. En primer lugar podemos decidirnos a hablar de los centauros tal y como se presentan en la mitología clásica, y así nuestro objeto llega a ser públicamente reconocible y localizable, pues tenemos que vérnoslas

con textos (verbales o visuales) en que se habla de centauros. Entonces se tratará de decir qué características ha de tener un ente de los que habla la mitología clásica para ser reconocido como centauro. En segundo lugar podemos intentar una indagación hipotética sobre las características que tendría que tener una criatura viviente en un mundo posible (que no es el real) para poder ser un centauro. En tal caso habríamos de definir las condiciones de subsistencia de este mundo posible advirtiendo que toda nuestra disertación se desenvuelve en el ámbito de esta hipótesis. Si nos mantenemos rigurosamente fieles a la empresa de partida, podemos decir entonces que nos ocupamos de un "objeto" que tiene alguna posibilidad de ser objeto de indagación científica.

Naturalmente este ejemplo es paradójico y no creo que nadie quiera hacer tesis sobre los centauros, sobre todo en lo que concierne a la tercera alternativa, pero me urgía mostrar cómo siempre puede constituirse un objeto de investigación públicamente reconocido en unas condiciones dadas. Y si se puede hacer con los centauros, otro tanto se podrá decir de nociones como el comportamiento moral, los deseos, los valores o la idea del progreso.

2.2.21 FORMATO LaTeX

El sistema **TeX** (se pronuncia [tej]) fue diseñado y desarrollado por **Donald Knuth** en la década del 70. Es un sofisticado programa para la composición tipográfica de textos científicos tales como artículos, reportes, libros, etc. **TeX** es en la práctica un estándar para publicaciones científicas en áreas como

matemática, física, computación, etc. LaTeX[7] es un conjunto de macros TeX preparado por **Leslie Lamport**. **LaTeX** no es un procesador de textos, es un lenguaje que nos permite preparar automáticamente un documento de apariencia estándar y de alta calidad. En general, solo necesitamos editar texto y algunos comandos y **LaTeX** se encarga de componer automáticamente la ***''formulería''*** del documento. A diferencia de un procesador de textos, con LaTeX tenemos un control más fino sobre cualquier aspecto tipográfico del documento **[Knuth, 84]**.

Hemos querido poner como efectividad de su manejo y obtención de óptimos resultados tipográficos escribiendo este proyecto y el borrador de Tesis, completamente en LaTeX, teniendo especial cuidado de poder trasladar el fuente a Linux y Windows bajo el formato UTF-8[8], como puede apreciarse el acabado tipográfico en la Figura 2.9 así como un artículo (paper) LaTeX en el ejemplo siguiente:

Figura 2.9: Vista del acabado tipográfico obtenido con LaTeX

[7] LaTeX : Además de LaTeX, existe otra opción, llamada ConTeXt. Este conjunto de macros TeX es menos famosa pero tal vez es más sencilla de usar y ofrece más posibilidades de edición TeX

[8] UTF-8 : Formato de mapa de caracteres UNICODE compatible entre las plataformas GNU/Linux, Mac y Windows

El sistema **TeX** (se pronuncia [tej]) fue diseñado y desarrollado por **Donald Knuth** en la década del 70. Es un sofisticado programa para la composición tipográfica de textos científicos tales como artículos, reportes, libros, etc. **TeX** es en la práctica un estándar para publicaciones científicas en áreas como matemática, física, computación, etc.

Ejemplo 2 (Un articulo básico con soporte UTF-8).

```
01:
02: \documentclass[10pt]{article}
03:
04: \usepackage[latin1,utf8x]{inputenc}  % tildes Win/Linux
05: \usepackage[spanish]{babel}          % idiomas es
06:
07: \usepackage{latexsym}
08: \usepackage{amssymb}
09:
10: \title{Resolución de Ecuaciones No-Lineales de Newton}
11: \author{Ph.D Armando Puentes Málaga}
12:
13: \begin{document}
14: \maketitle
15:
16:  \section{Presentación}
17:      Las ecuaciones no-lineales son parte de las ecuaciones
18:  diferenciales, por lo que el método de resolución mas ....
19:
20: \section{Resultados}
21:      Hemos obtenido de:
22:
23: $$
24:  \sum_{i=1}^{N} f(x) = \int_{a}^{b} \frac{5x}{x^2+3}  \; dx
25: $$
26:
27: \end{document}
28:
```

LaTeX[9] es un conjunto de macros TeX preparado por **Leslie Lamport**. **LaTeX** no es un procesador de textos, es un lenguaje que nos permite preparar automáticamente un documento de apariencia estándar y de alta calidad. En

[9] LaTeX : Además de LaTeX, existe otra opción, llamada ConTeXt. Este conjunto de macros TeX es menos famosa pero tal vez es más sencilla de usar y ofrece más posibilidades de edición TeX

general, solo necesitamos editar texto y algunos comandos y **LaTeX** se encarga de componer automáticamente la ***''formulería''*** del documento. A diferencia de un procesador de textos, con LaTeX tenemos un control más fino sobre cualquier aspecto tipográfico del documento **[Knuth, 84]**.

Resolución de Ecuaciones No-Lineales de Newton

Ph.D Armando Fuentes Málaga

20 de agosto de 2011

1. Presentación

2. Resultados

Figura 2.10: Resultado PDF después de compilar con pdflatex.

2.3 MARCO CONCEPTUAL

2.3.1 ANTIVIRUS

Es el software especializado en la detección y eliminación de software malicioso potencialmente peligroso para el sistema operativo host, diversas empresas están especializadas en el desarrollo de estos productos de alta seguridad, las empresas peruanas más respetadas de los años 80s fueron PER Antivirus y The Hacker.

2.3.2 HEURÍSTICA

Es el proceso de resolución de problemas por medio de la experiencia, para el caso de un software puede usarse una base de datos para representar las reglas de conocimiento.

El principio de la heurística es muy sencillo y fácil de comprender. Otro tema bien distinto es que los algoritmos resultantes sean mejores o peores, que es donde está realmente la cuestión. Imaginemos que vamos por la calle y notamos que nos sigue alguien. Nos fijamos por un reflejo o nos damos la vuelta y ver que ese alguien tiene un aspecto sospechoso y actúa de forma sospechosa. Lo más normal es que en este punto, clasifiquemos a ese individuo como sospechoso y cambiemos el estado de tranquilidad por un estado de alerta. Eso es Heurístico, es un comportamiento basado en la experiencia, que se va modificando en función a los acontecimientos.

2.3.3 PROACTIVIDAD

Es la técnica de resolución de problemas basada en la experiencia y mejorada por la capacidad de adelantarse a una amenaza, compleja de codificar pero eficaz en la resolución de problemas. La Proactividad es tomar la iniciativa pero tener que asumir la responsabilidad de las acciones realizadas.

2.3.4 MALWARE

Es el conjunto con el que se denomina a los virus, gusanos, troyanos y cualquier otro tipo de software desarrollado para crear estragos informáticos en una computadora o propagarse en una red de ellas, además de tener la capacidad de auto replicarse, asegura su existencia a través de la reproducción.

2.3.5 DATOS E INFORMACIÓN

El dato es el detalle exacto sobre algún tema, mientras que la información puede catalogarse como un conjunto de datos, es decir actúan en singular y plural para contener información sobre determinados temas.

Para nuestra investigación, tomamos a la información de los usuarios como el elemento más importante a cuidar, para ello un antivirus que sea capaz de proteger su información impidiendo la mayor cantidad de infiltraciones de las diversas amenazas informáticas.

2.3.6 SOCKETS

Usualmente encapsulados en alguna clase que permitirá la conexión entre aplicaciones que pueden estar dispuestos local o remotamente y que tengan en común una red, una IP y un puerto para poder comunicarse, el protocolo más popular de conexión es el TCP que permite una verificación de la conexión, además de la posibilidad de verificar la integridad de la información que enviamos, evitando de este modo la pérdida de información que pueda conllevar a una comunicación errada.

2.3.7 ACTUALIZACIÓN POR INTERNET

Sin ir muy lejos debemos tener en cuenta que la Internet no es más que una red que aglomera sub-redes dentro de ella todas ellas direccionadas a través de ***Routers, Switch o contraladores de Redes*** corporativas además de la red de servidores de dominio y los servidores de nuestro proveedor local de Internet.

Podemos descargar archivos no necesariamente conectados a Internet, incluso en una red local o LAN es perfectamente posible, solamente hacemos uso de algún protocolo de comunicación como **TCP/IP** o **FTP** para poder intercambiar archivos o datos. Para nuestro caso tomamos el término Actualización por Internet como solución a grandes distancias y poder hacer accesible la información a personas que no se encuentran en una LAN cercana a la que estemos trabajando.

2.3.8 RAMDRIVE

Son los conocidos como discos en memoria RAM, la característica principal es que son rápidos para acceder, sencillos de implementar pero sobre todo la información que se almacena es volátil, puesto que por defecto las operaciones de escritura sobre tal no tienen efecto sobre el disco duro físico. Son ideales para implementar aplicaciones que requieran acceso a gran cantidad de información que no podría entrar en un disco flexible de 1.44 MB ni podrían ser usado con facilidad en un CD-ROM por ser únicamente de lectura, para el caso de los dispositivos de almacenamiento USB, la velocidad sigue siendo lento, en las versiones USB 1.0 y USB 2.0.

2.3.9 COMPRESIÓN Y ENCRIPTACIÓN

Es la técnica computacional que permite reducir el tamaño de grandes cantidades de información en cantidades de fácil portabilidad, el caso práctico siempre es para las imágenes de mapas de bits que son las que mayor cantidad de información ocupan, para ello se mapea todo el bloque de información se busca patrones que repitan y puedan ser catalogados como elementos padre y a su vez a partir de ellos agregar nodos o hijos generando de este modo todo un árbol de información optimizado al máximo.

La compresión de la información tiene que ver también con el cifrado de información, pues al usar un patrón de comparación es posible agregarle otro patrón conocido como llave (Key), la que permitirá cifrar la información bajo

un componente o patrón nuestro, como una clave secreta que permita mantener a salvo la información, sobre todo de crackers que deseen alterar nuestros datos.

2.3.10 DISPOSITIVO AUTO-ARRANCABLE

La palabra técnica es **autoboot**, se refiere a la capacidad de poder auto ejecutarse sobre una computadora sin la necesidad del sistema operativo host (instalado por defecto), en ejemplo general a esto fueron los ***floppy disk*** de MS-DOS, los Live CD, DVD y ahora los Dispositivos USB auto-arrancables.

Generar un auto-arrancable es relativamente sencillo, partiendo desde cero únicamente es necesario seguir con la secuencia de arranque del BIOS (Basic Input Output System) del computador el cual al encender el computador se ejecuta una rutina conocida como ***bootstrap*** que salta al sector **7C00:0000**, donde enganchamos una rutina de carga usualmente de 512 bytes, escrita en algún ensamblador de calidad como **TASM, NASM, FASM**, etc.

Es posible crear USB arrancables, Floppy disk arrancables e incluso CD-DVD arrancables con herramientas de uso común como MagicISO, ISO-Master, HP USB Toolkit entre otros, por lo que nuestra tarea se facilita de manera que podemos concentrarnos en obtener un producto versátil sobre cualquier dispositivo de almacenamiento.

2.3.11 LIVE ANTIVIRUS

Es el término que acuñaremos a la capacidad de que un Software Antivirus sea capaz de ejecutarse sin la necesidad de un sistema operativo, sino soportado por un controlador básico de entrada y salida, que permita cargar las unidades de disco virtual, disco duro y dispositivos de almacenamiento USB, soporte básico de teclado y una interfaz gráfica sencilla y amigable.

2.3.12 MODO PROTEGIDO .386

Cuando un computador inicia y carga el **bootstrap**, siempre inicia en **Modo Real**, lo que significa que ejecuta instrucciones de 16 bits y una memoria segmentada y desplazamientos usualmente vistos como **FFFF:0000**, unidos por *Segment* and *Offset*, lo que nos dice que podemos usar una memoria teórica de 1 MB, por que el resto se usa para el sistema operativo host *(juego de instrucciones **WORD** de 16 bits **AX, BX, CX, DX, STOSB, STOSW**)*.

El **Modo Protegido** usualmente marcado como 386p, se logra al cambiar el registro de control **CR0** y enganchar manualmente la tabla de interrupciones **IRQ**, además de iniciar las tablas de los descriptores GDT, LDT e IDT. Pero después de este complicado paso puede usarse una memoria teórica de 4 GB por programa que se ejecuta ya que además puede usarse el modo compartido de ejecución o **Multitasking**, sobre el que corren la mayoría de sistemas operativos (juego de instrucciones **DWORD** de 32 bits **EAX, EBX, ECX, EDX, STOSD, LODSD**).

Actualmente con los procesadores de 64 bits existe el **Modo Extendido** o **Long Mode**, que consta básicamente de un direccionamiento de memoria de 18-HexaBytes, además de la capacidad de poder trabajar con varios procesadores de forma paralela y un set de instrucciones completamente funcional (juego de instrucciones **QuadWORD** de 64 bits **RAX, RBX, RCX, RDX, STOSQ, STOSQ**).

2.3.13 META COMPILADO

Es una técnica de programación que nos permitirá unir librerías escritas en **Assembler** o lenguaje Ensamblador (Para obtener mejor velocidad y poco código), para ser unidas con archivos fuente escritos en **C++**, (debemos aclarar que esta técnica se usa bastante en la codificación de compiladores y sistemas operativos). La librería contendrá funciones como la que detallamos en:

Ejemplo 3 (Ejemplo de Funcion de Libreria de Memoria).

```
.386p
.Model Flat

.CODE
  PROC kSetMemDWord ( DWORD pSegment, DWORD lVal, DWORD lCont )

     PUSH BP
     MOV  BP, SP          ; StackFrame
     ;--------------------------------
     MOV EAX, pSegment
     MOV ES , EAX
     MOV DI , 0
     MOV ECX, lCont
     MOV EAX, lVal
     REP STOSD
     ;--------------------------------
     POP  BP
  ENDP
END
```

Ejemplo 4 (Ejemplo Llamada en C++).

```
// acceder a la VGA de texto
for( int i=0; i<nBanks; i++ )
{    kSetMemDWord ( 0xB800, 0x0000, 0xFFFF );  }
```

Ejemplo 5 (Secuencia de Compilado y Enlazado).

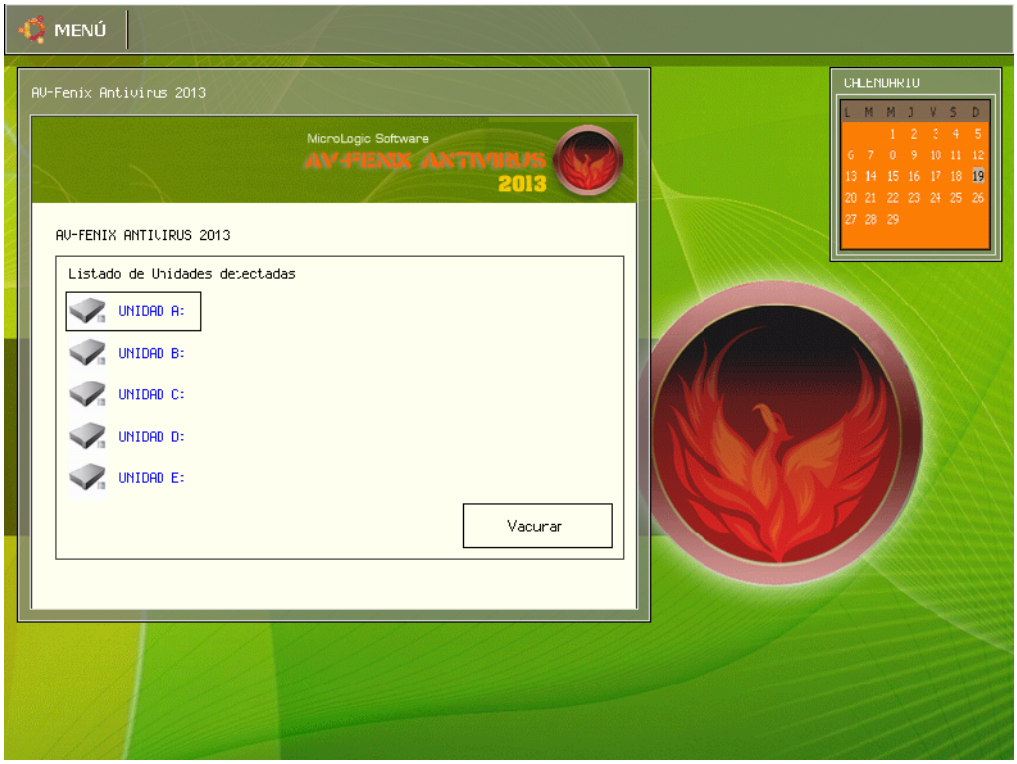
```
BCC programa.cpp -o programa.obj
TASM libreria.asm
tLink programa.obj libreria.obj
```

Solo se requiere que al compilar se haga uso del enlazador (Linker) de forma que no use las librerías estándar o CRT (Common RunTime), dejando el compila del código C++ sin enlazar, luego compilar manualmente los archivos .ASM y finalmente enlazar todos los archivos OBJ.

2.3.14 INTERFAZ GRÁFICA DE USUARIO

Son los elementos gráficos que permiten que el usuario interactúe fácil e intuitivamente con un programa, para nuestro caso de estudio se ha basado la GUI (Graphical User Interface) en la librería desarrollada para el proyecto **Kronoz OS - KrystalView Desktop**, la cual es un escritorio basado completamente sobre el estándar de video **VESA** (Video Especification Standar Asociation) que aparece en 1994 con el estándar **SuperVGA** que permitía acceder a mas de 256 colores, con calidades de video de 15bits, 16bits, 24bits y actualmente de 32bits a una gama de 24 millones de colores como es el caso de la implementación de la librería que hemos usado (Puede verse en sitio web del desarrollador *filicc.tk* o directamente en:

http://filicc.webcindario.com/FILICC/cntArticles/ContListArticles.php?Ident=2, bajo el nombre de KronozVESA 0.2.3.



CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. METODOLOGIA Y PROCEDIMIENTO

La seguridad informática que requiere este tipo de investigación ha permitido que elijamos de entre metodologías de desarrollo rápido y flexible, para poder concentrar la mayor cantidad de esfuerzos en el desarrollo iterativo y en constante prueba y obtención de resultados.

3.1.1. METODOLOGIAS TRADICIONALES

Son procesos basados en la documentación y el orden para conseguir productos de software de calidad, a lo que los que desarrolladores de software que no les gustan del papeleo lo han denominado metodologías burocráticas por ocupar el mayor tiempo en la documentación y quitando el encanto a la programación.

3.1.2. METODOLOGIAS ÁGILES

Las metodologías ágiles se han desarrollado con mucho éxito en los últimos años, empujan hacia una organización menos formal y jerárquica en el desarrollo de software y más centrada en la persona, con un énfasis mayor en el objetivo principal, eliminar actividades que se relacionaron con algunos documentos 'formales' de especificaciones que no tienen una relación directa clara con el resultado final del producto, de entre las metodologías ágiles, la que goza de mayor popularidad es la programación extrema, propuesta en 1999 por **Kent Beck**, en un libro titulado precisamente "abraza el cambio". La programación extrema recibe este calificativo precisamente porque defiende un enfoque radical. Reconoce las bondades de las prácticas de las metodologías tradicionales y propone llevarlas hasta su extremo: ***"si diseñar es bueno, diseñemos todo el tiempo"***, ***"si las pruebas son buenas, probemos todo el tiempo"***.

3.1.3. METODOLOGIAS XP (eXtreme Programming)

En los tiempos en que la informática empezó a hacerse popular, en el que sólo había pantallas de texto, no había entornos de ventanas y las impresoras usaban papel con agujeros a los lados y rayas horizontales, las aplicaciones eran bastante más sencillas que las actuales. Bastaba uno o dos programadores, que totalmente a su aire y en un plazo razonable de tiempo eran capaces de hacer programas útiles **[Kent, 05]**.

La programación extrema, lejos de ser un proceso incontrolado, es una metodología muy disciplinada y que se apoya en cuatro valores fundamentales:

A) **Comunicación :** Se hace énfasis en que la comunicación, para ser efectiva, debe involucrar a todos los participantes en el proyecto, y debe ser libre y sincera.

B) **Simplicidad :** Nunca debe perderse de vista que el objetivo de un proyecto es proporcionar valor al cliente; no es demostrar la pericia técnica del equipo ni construir una aplicación que resuelva más problemas que los del cliente.

C) **Retroalimentación :** No se puede dirigir adecuadamente un proceso si no se dispone de realimentación permanente sobre su progreso. La realimentación puede provenir del cliente, de los programadores, de herramientas automáticas.

D) **Coraje :** A veces, hacer lo que es correcto requiere valor. Por ejemplo, hay que tener coraje para reescribir código que funciona pero ha alcanzado su límite de escalabilidad.

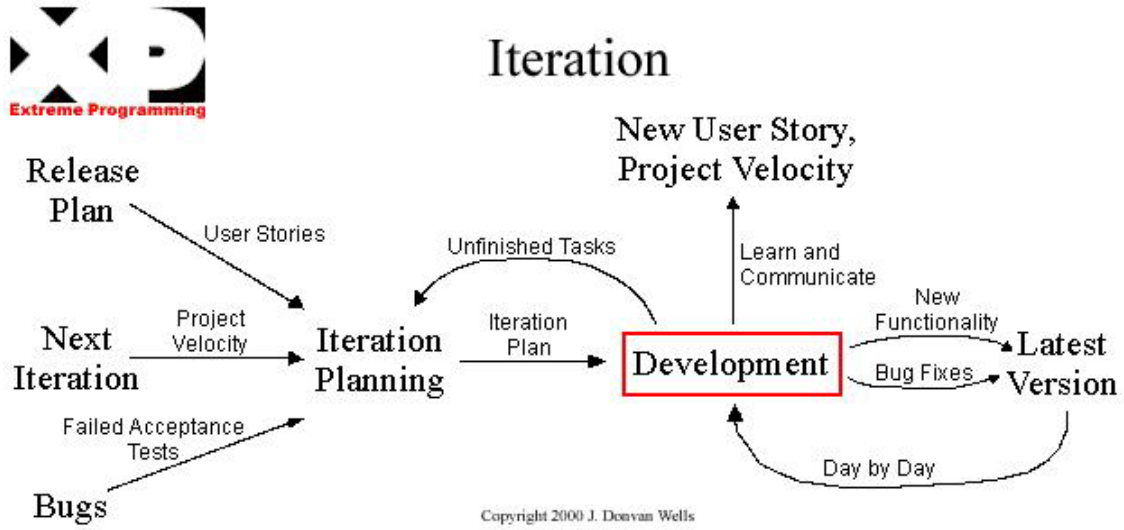


Figura 3.1: Esquema Grafico de la Metodología XP

Estos cuatro valores dan origen a cinco principios básicos: conseguir una realimentación rápida, no complicar las cosas con suposiciones (asumir que las cosas son simples), realizar cambios incrementales, abrazar el cambio y generar productos de calidad. Los cinco principios se manifiestan a través de las prácticas de la programación extrema.

3.2. MODELAMIENTO CON U.M.L.

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa

\textbf{Rational Software Co}. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas "guerras de métodos" que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. En la figura superior se puede ver cuál ha sido la evolución de UML hasta la creación de UML 1.3, en el que se basa este documento. Hay que tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación. En estos conceptos se sigue el método propuesto por Craig Larman **[Larman, 99]** que se ajusta a un ciclo de vida iterativo e incremental dirigido por casos de uso.

3.3. METRICAS DE VALIDACIÓN DE SOFTWARE

3.3.1. METRICA ISO/IEC 9126

ISO 9126 es un estándar internacional para la evaluación de la calidad del software. Está reemplazado por el proyecto SQuaRE, ISO 25000:2005, el cual sigue los mismos conceptos.

El estándar está dividido en cuatro partes las cuales dirigen, respectivamente, lo siguiente: modelo de calidad, métricas externas, métricas internas y calidad en las métricas de uso y expendido.

El modelo de calidad establecido en la primera parte del estándar, ISO 9126, clasifica la calidad del software en un conjunto estructurado de características y sub-características.

CAPÍTULO IV

ANÁLISIS Y PRESENTACIÓN DE RESULTADOS

4.1. ANÁLISIS DE REQUISITOS

REQUISITOS

- Computador Laptop Portatil **Lenovo Thinkpad T400**.
- Sistema de **Backups** en Discos Portátiles USB de **1.TB** (Terabyte).
- Sistema Operativo **GNU/Linux** Personalizado **FINESI-BUNTU 11+**.
- Escritorio **GNOME 4**
- Emuladores x86 y maquina virtual **Oracle VirtualBox 4, QEMU**.
- Emuladores ARM (con **Android OS 2.3+**)
- Windows XP 32bits virtualizado para **VirtualBox.**
- Windows XP 64bits virtualizado para **VirtualBox**.
- Entorno de Desarrollo Integrado IDE Code::Blocks 10 Win/Linux
- Librería de clase Cross-Plattform **wxWidgets 2.8.6+**
- Editor GEDIT, NANO para programas en C/++ y NASM.

- Ensambladores NASM, TASM (Turbo Assembler).
- Compilador Borland C++ 3.5 para Windows.
- Entorno de Desarrollo Integrado Microsoft Visual C++ .NET 2005
- Installshield para Visual C++ (Para crear el Instalador).
- Depuradores Turbo Debugger 5, OllyDBG y Microsoft Debuger.
- GIMP y/o Fireworks para las imágenes y logos.
- 4 Paquetes de 500 Hojas (2000) para la documentación.
- 50 DVDs para distribuir la versión 2013.
- Pago de **$/. 12.00** por la compra del dominio **www.av-fenix.com**

CARACTERÍSTICAS CONSEGUIDAS

- Interfaz Gráfica de Usuario elegante pero intuitiva.
- Scanner Antivirus actualizado a **Enero de 2013**.
- Base de datos de firmas de virus y gusanos a **Enero 2013**.
- Agente de Actualizaciones para descargas Manuales
- Agente de Actualizaciones al detectar conexión a Internet.
- Soporte multiplataforma Windows, GNU/Linux, Finesi-Buntu 11+
- Modo **LiveAntivirus** en CD/DVD y USB.
- Soporte para FAT, FAT32 y NTFS.
- Soporte para Descargas y Actualizaciones.
- Soporte On-Line a los usuarios desde: **http://www.av-fenix.com**

4.2. DISEÑO

4.2.1. DIAGRAMA DE CLASES

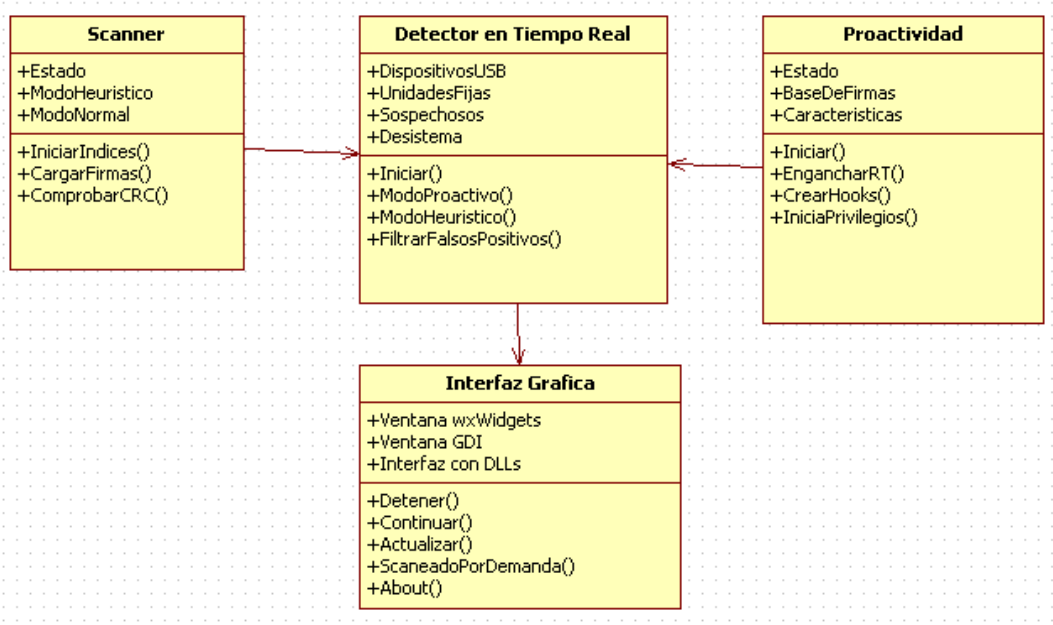


Figura 4.1: Diagrama de clases

Aquí se observa la dependencia de la clase Interfaz Grafica de las anteriores clases justamente para separar el trabajo, simplificar las operaciones y sobre todo poder hacer mantenimiento en un módulo sin que afecte al anterior, solo en su dependencia. Las clases que administran el trabajo en tiempo real y la Proactividad se muestran descriptivamente, mientras que en la codificación real contiene llamadas a librerías DLL ***(Librerías de Enlace Dinámico)*** entre otras. Se ha separado el código en DLLs para poder administrarlas en proyectos distintos, de este modo no afecte las modificaciones de una librería sobre la otra.

4.2.2. DIAGRAMA DE SECUENCIA

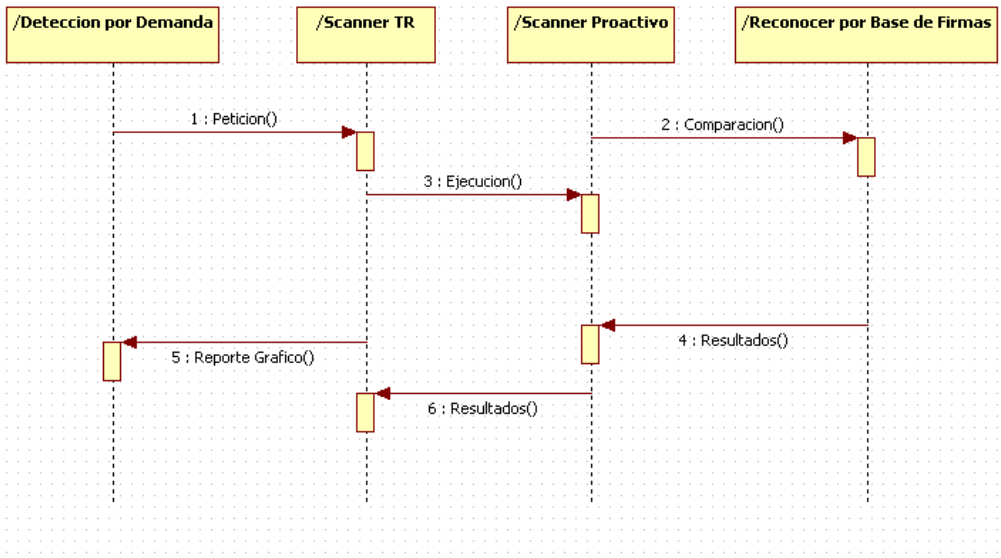


Figura 4.2: Diagrama de secuencia

El usuario de la aplicación únicamente puede pedir una escaneado por demanda, mientras tanto el motor proactivo en tiempo real realiza las comprobaciones de los dispositivos extraíbles ya sean memorias USB, memorias SD de teléfonos celulares, cámaras fotográficas y hasta discos duros extraíbles. Típicamente se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se dispone de la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos.

4.2.3. DIAGRAMA DE ACTIVIDADES

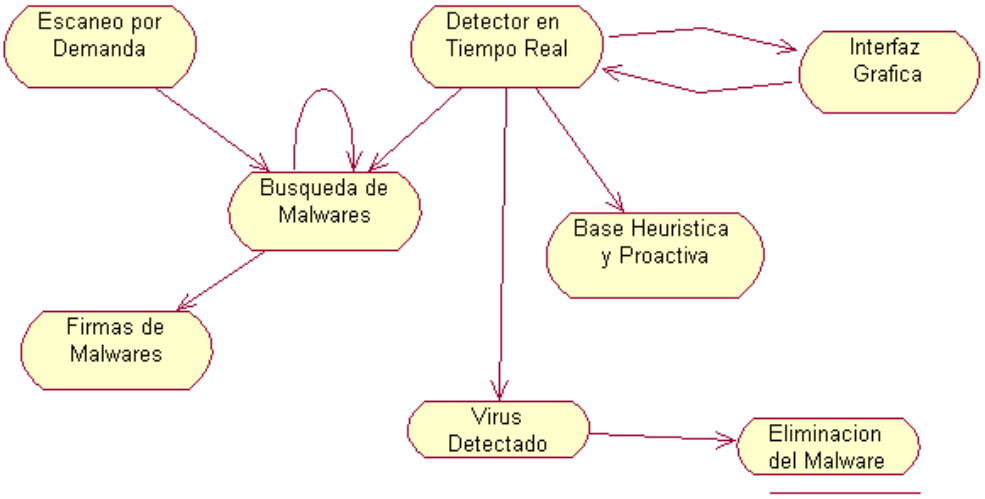


Figura 4.3: Diagrama de actividades

4.3. CODIFICACIÓN E IMPLEMENTACIÓN

4.3.1. DESARROLLO DE LA VERSION WIN32

A) VERSION BASADA EN M.F.C.

Librería de Clases Microsoft, está presente para los compiladores Visual C++ 1.0 hasta la actualidad, es una versión que encapsula la **API Win32** mediante un esquema de derivación de clases, como **CWinApp**, **CFrameWnd**, **CFrameView**, **CDialog** basadas todos en la clase \verb|CWnd| la que permite derivar en componentes visuales como Ventanas, Botones, Listas, Cuadros de control, etc.

La codificación está completamente escrita en C++ y requiere conocimientos de Programación Orientada a Objetos para poder escribir un programa M.F.C. Por ejemplo una aplicación M.F.C. básica se vería del siguiente modo:

Ejemplo 6 (Programa fuente de aplicación MFC).

```
01:
02: #include <afxwin.h>
03:
04: class CMyApp : public CWinApp
05: {
06:   public:
07:     virtual BOOL InitInstance();
08: };
09:
10: BOOL CMyApp::InitInstance()
11: {
12:     AfxMessageBox( "Demo MFC sobre Windows" );
13:     return TRUE;
14: }
15:
15: CMyApp theApp;
17:
```

Figura 4.4: Captura de AV-Fenix 2013 para Windows

Véase el código fuente versión Win32 en el **Anexo A**

B) VERSION BASADA EN wxWidgets

Desarrollado por Julian Smart **[Jsmart,06]** con la primera version en 1992 es aun mas antigua que M.F.C. pero la implementación sigue en desarrollo y completamente funcional además de ser **Cross-Plattform** (inter sistemas operativos e inter compiladores), las librerías de compilación están escritas en C++ especialmente para el compilador GCC pero la portabilidad de sus APIs tiene implementaciones en lenguajes Python, Perl y Java.

A diferencia de **MFC** las librerías **wxWidgets** están bajo la Licencia GPL (Licencia Publica General) lo que permite que terceros accedan al desarrollo y mejora constante.

Ejemplo 7 (Programa fuente de aplicación wxWidgets).

```
01: #include <wx/wx.h>
02:
03: class CMyApp : public wxApp
04: {
05:   public:
06:     virtual bool OnInit();
07: };
08:
09: bool CMyApp::OnInit()
10: {
11:     wxMessageBox( _("Demo con: wxWidgets sobre Linux") );
12:     return true;
13: }
14:
15: IMPLEMENT_APP(CMyApp);
```

Figura 4.5: Captura de AV-Fenix 2013 version wxWidgets, compilado con MingW GCC 4.2 obteniendo el mismo resultado.

4.3.2. DESARROLLO DE LA VERSION GNU/LINUX

A) REQUERIMIENTOS

- GNU/Linux Finesi-Buntu 11+
- Compilador GCC 4.2
- Librería de clase wxWidgets 2.8.2
- Entorno de Desarrollo Integrado IDE Code::Blocks 10
- Diseñador GUI wxForms para C++
- Editor gráfico GIMP e InkScape
- Editor GEDIT para escribir C/C++

B) CODIFICACION

Se escribió en el IDE Code::Blocks sobre escritorio GNOME y la librería de clases wxWidgets 2.8.6, se muestra en la parte inferior parte la codificación.

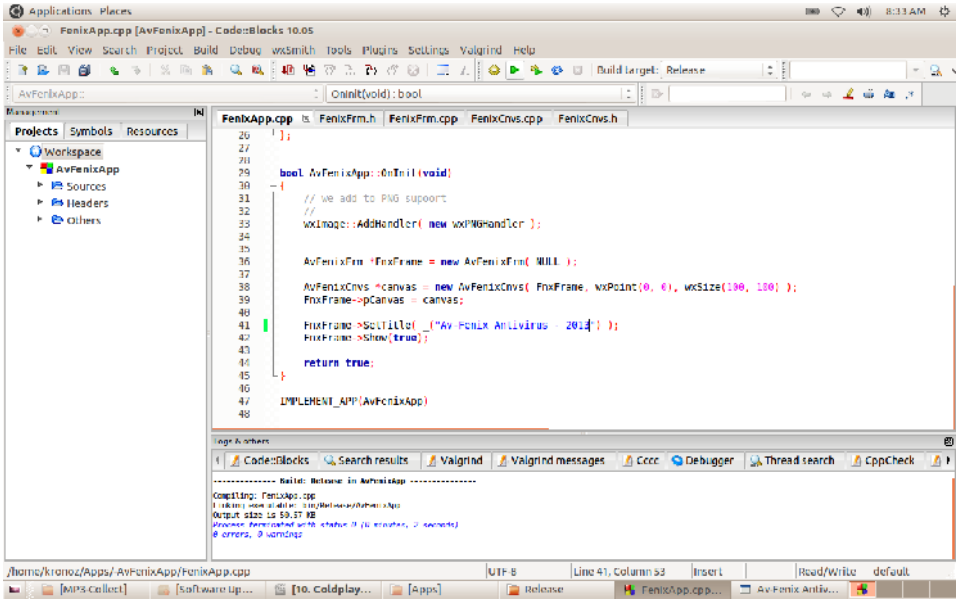


Figura 4.6: Captura del IDE Code::Block 10 sobre GNOME.

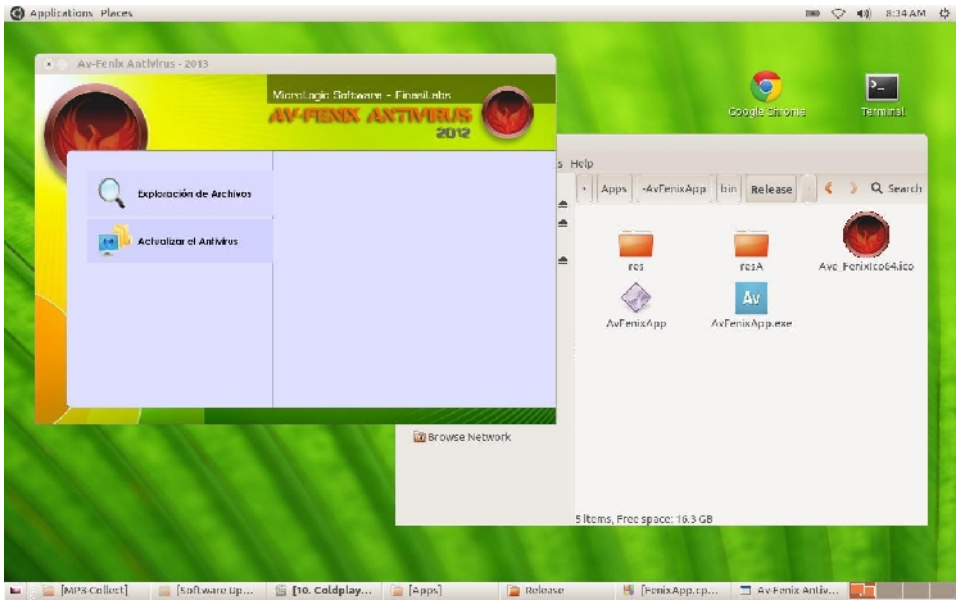


Figura 4.7: Resultado de compilación y ejecutando un binario ELF.

4.3.3. DESARROLLO DE LA VERSION ANDROID

A) REQUERIMIENTOS

- GNU/Linux **Finesi-Buntu** o **Windows**
- IDE Eclipse para Java Win/Linux
- Android SDK, ADV Manager
- Emulador para procesador ARM
- Editor gráfico GIMP e **InkScape**

B) CODIFICACION

El sistemas operativo **Android OS** desarrollado actualmente por la empresa **Google Inc**. es uno de los mas populares y versátiles para los SmartPhones o teléfonos inteligentes, debe tenerse en cuenta que el desarrollo no difiere mucho de las **APIs** de Java como **Swing** o **JSon**.

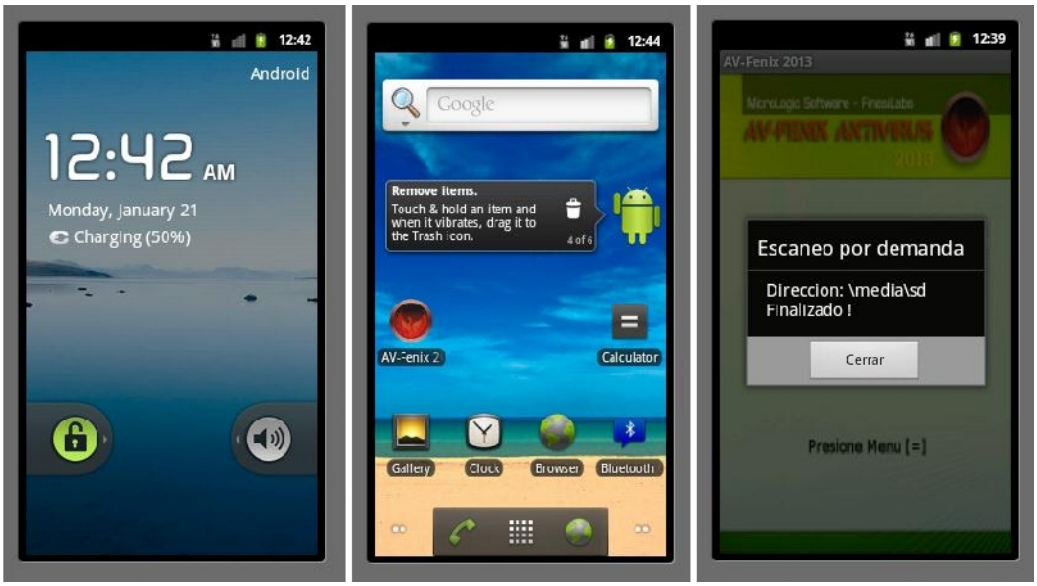


Figura 4.8: Smartphone con Android OS y AV-Fenix.apk ejecutandose.

Ejemplo 8 (Programa básico con Android SDK para Java).

```
01: import android.app.Activity;
02: import android.os.Bundle;
03:
04: public class MainActivity extends Activity {
05:     public void onCreate(Bundle savedInstanceState) {
06:         super.onCreate(savedInstanceState);
07:         setContentView(R.layout.activity_main);
08:     }
09: }
```

4.3.4. DESARROLLO DE LA VERSION LIVE-ANTIVIRUS

Esta es la versión mas prometedora en el desarrollo del Antivirus se proyecta dar soporte a los computadores aun cuando no haya un sistema operativo ya sea por daños o accionar de **Malwares**, el modo **LiveDVD** permitirá vacunar en modo activo.

A) REQUERIMIENTOS

- Sistema Operativo Windows
- Compilador Borland C++ 3.1
- Turbo Assembler 5.1
- Turbo Linker 5.1
- Librería gráfica VESA **KrystalView** (filicc.tk)
- Emulador **VirtualBox 4**

B) RESULTADOS

Una vez compilado el software carga por defecto un Agente de Video que se encarga de los modos de video y elige los mas óptimos, esto debido a que cada marca de tarjeta de video tiene un modo y resoluciones compatibles pero no estandarizados lo que complica el uso del modo grafico pero gracias a la librería **KrystalView** y el Agente de Video esto se simplifica y siempre podremos obtener 32BPP o 24 Millones de colores en resolución mínima de 640x480.

Véase código fuente de **Kronoz VideoAgent** en **Anexo C**.

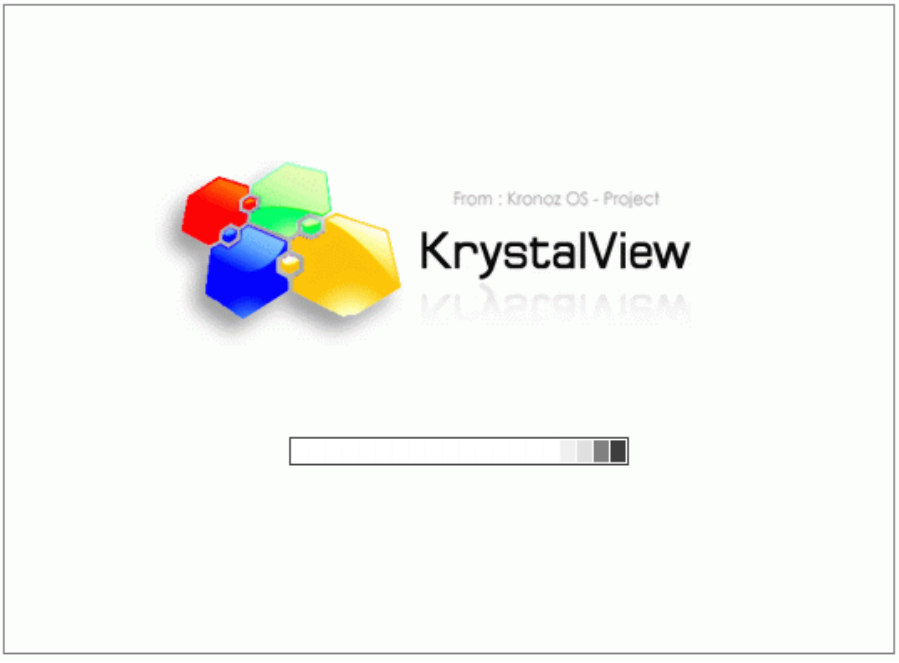


Figura 4.9: Librería VESA de 32BPP KrystalView

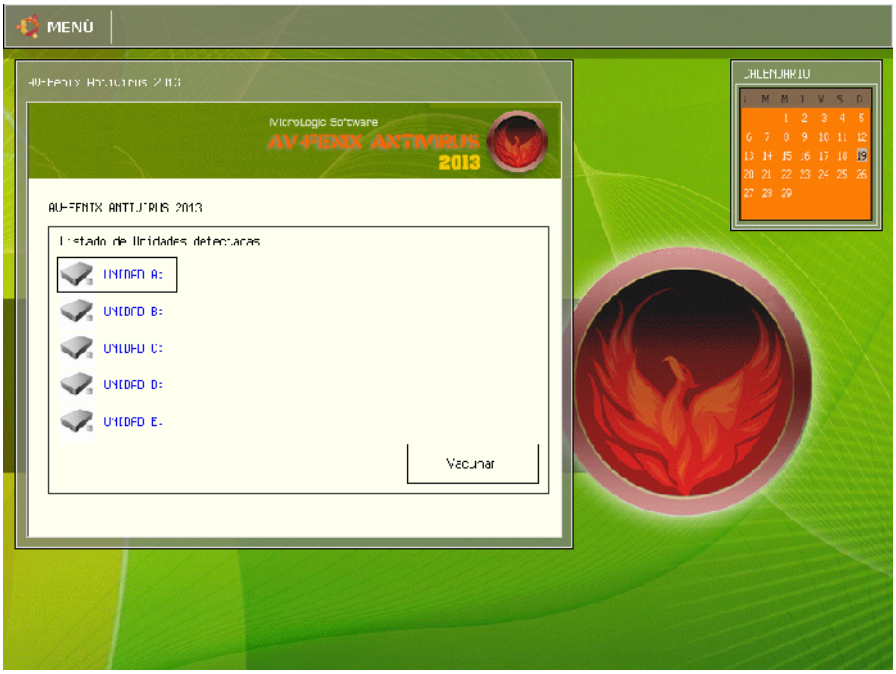


Figura 4.10: Vista de AV-Fenix 2013 LiveDVD sobre *KrystalView*

Ejemplo 9 (Uso de librería VESA KrystalView en AV-Fenix).

```
/*********************************************************
 * Kronoz Project - Av-Fenix Antivirus
 * Modulo : KrystalView Environment + [KronozVideo Agent]
 *
 * Compilar : bcc -ml -B -3 kvFenix.cpp
 *********************************************************/

#include "kvGraph.h"
#include "kvFonts.h"
#include "kvWidgts.h"

class KvFenixApp : public KApp {
    private:
        BOOL OnInit();
        void OnExit();
};
```

4.3.5. DESARROLLO DEL SITIO WEB Y LA ACTUALIZACION

A) REQUERIMIENTOS

- GNU/Linux **Finesi-Buntu**
- Servidor Web Apache LAMPP, XAMPP.
- Editor Goldfish para HTM y PHP.
- Editor gráfico GIMP e **InkScape** para logos.

B) CODIFICACION

Se ha desarrollado módulos que encapsulan las partes esenciales de un contenido HTML y agregando modularidad al código así poder re-usar código que fácilmente puede ser agregado como librería.

```
1
2   <?PHP
3        include( "fnxModulos/cmsHeader.php" );
4   ?>
5
6   <div class=”mainView”>
7
8    <h2> Presentación </h2>
9
10  <p Align=Justify>
11  Presentamos la nueva versión <b>AV-Fenix
12  Antivirus 2003</b> con mejoras en el Scanner
13  nuevos soportes y la nueva versión <b>LiveDVD</b>
14  para <b>Vacunas en Pasivo</b>. Disfruten este
15  producto y dejen sus comentarios. Gracias.
16  </p>
17
18  <p Align=Right>
19    El equipo de Desarrolladores <br>
20    <b>FinesiLabs y MicroLogic Software</b>
21  </p>
22
23  </div>
24
25  <?PHP
26       include( "fnxModulos/cmsFooter.php" );
27  ?>
28
```

C) CONTENIDO DEL SITIO WEB

- **Presentación :** Contenido textual sobre el producto AV-Fenix Antivirus 2013.
- **Descargas :** Están disponibles las actualizaciones de la base de firmas así como la versión para Win32, Linux, Android y LiveDVD.
- **Comentarios :** Se desarrollo un Log (historial) para poder mantener comunicación con los usuarios que adquieran el producto o deseen mas actualizaciones.
- **Contactos :** Información sobre los creadores del producto.

.

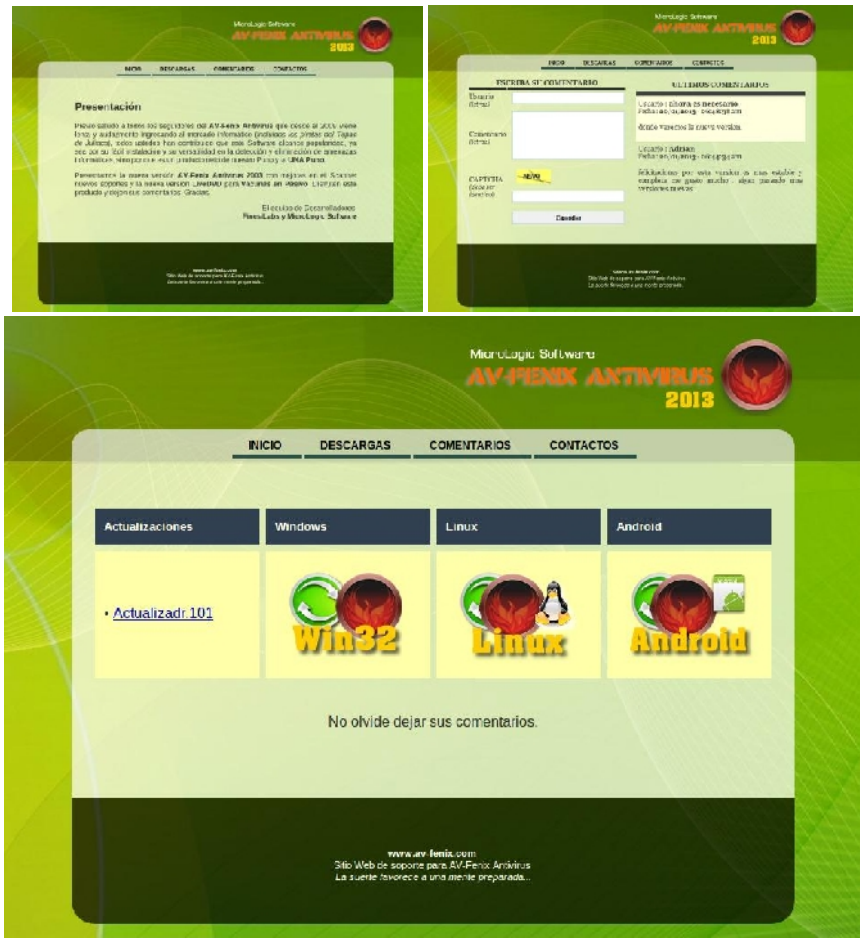


Figura 4.11: Vista del sitio WEB disponible en: www.av-fenix.com

4.4. PRUEBAS

4.4.1. PRUEBAS DE DESEMPEÑO

CASO 1: PLUG AND PLAY DETECTION

Cuando se conecta un dispositivo de almacenamiento extraíble en el computador el sistema operativo queda vulnerable por algunos segundos es en este instante que nuestro Antivirus actúa sobre el dispositivo realizando una vacuna rápida sobre la raíz de directorios restaurando información y/o limpiando de amenazas potenciales.

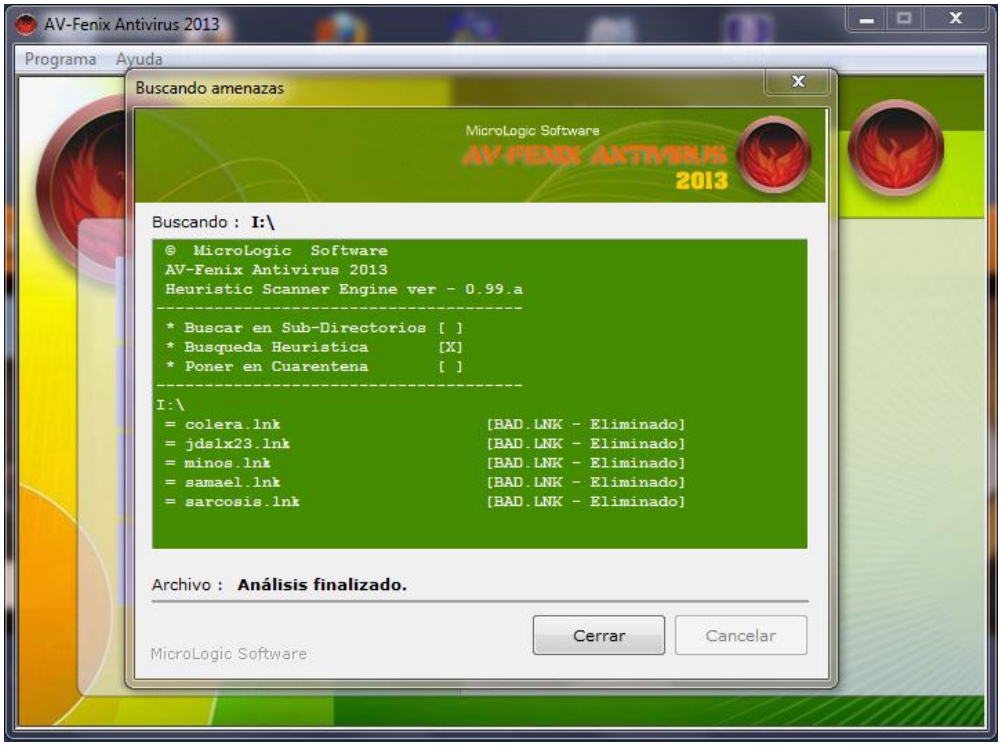
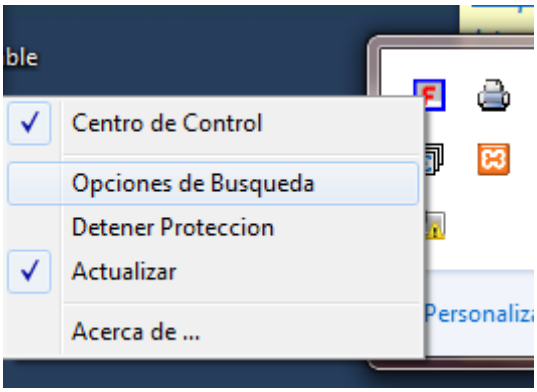


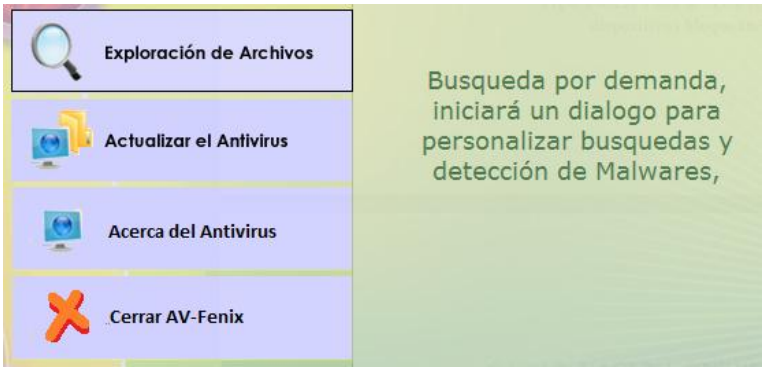
Figura 4.12: Vista de AV-Fenix 2013, detector y vacunador de nuevos dispositivos bloqueando InSitu un 80% de infiltraciones.

CASO 2: VACUNA ONDEMMAND

La vacuna **OnDemmand** se le denomina a la vacuna por petición explicita del usuario, cuando requiere una evaluación más exhaustiva.



Modo 1: Por la barra de tareas, click derecho y menú contextual.



Modo 2: Por el Panel de Control, menú AV-Fenix.

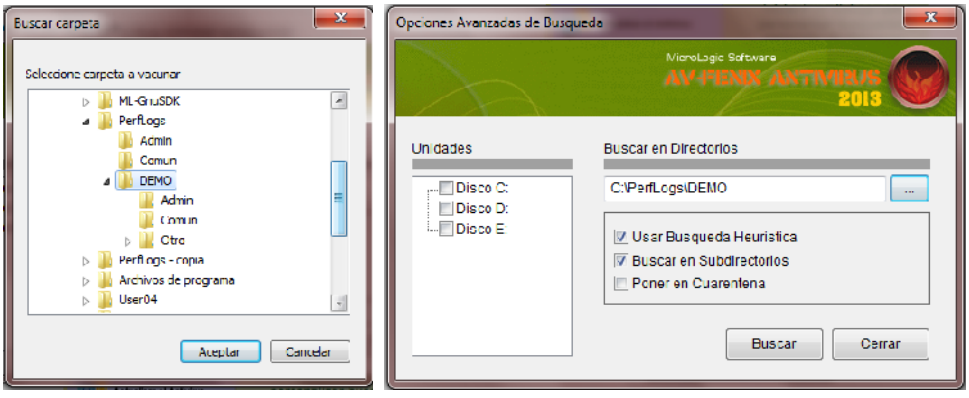


Figura 4.13: Dialogo de selección de Unidad e incluso sub-directorios.

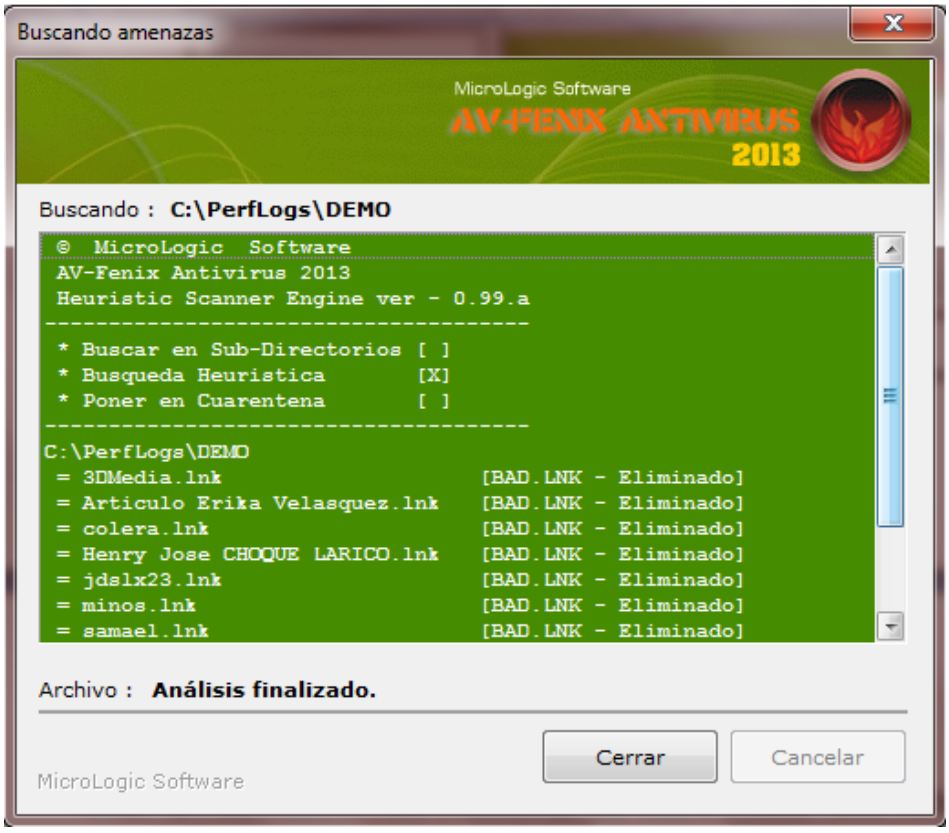


Figura 4.14: Resultados de la vacunación OnDemmand.

Para la muestra se infecto bajo control la carpeta **C:\PerfLogs\DEMO** con 14 variedades de gusanos y dañado 4 subdirectorios, después de la vacuna **OnDemmand** tenemos los siguientes resultados.

Infección Controlada	Infectados	A Prueba	Limpiados
Gusanos	14	14	13
Carpetas Dañadas	4	4	4
%	--	**100%**	**90%**

4.4.2. EVALUACIÓN DE CALIDAD DE SOFTWARE - ISO 9126

El modelo de calidad establecido en la primera parte del estándar **ISO 9126**, clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas.

Para la evaluación del nivel de calidad del Sistema Antivirus se aplico los indicadores de calidad del Estándar **ISO – 9126**, para lo cual se hizo uso de la guía de evaluación de calidad de software que describe los factores que se muestra en el gráfico siguiente:

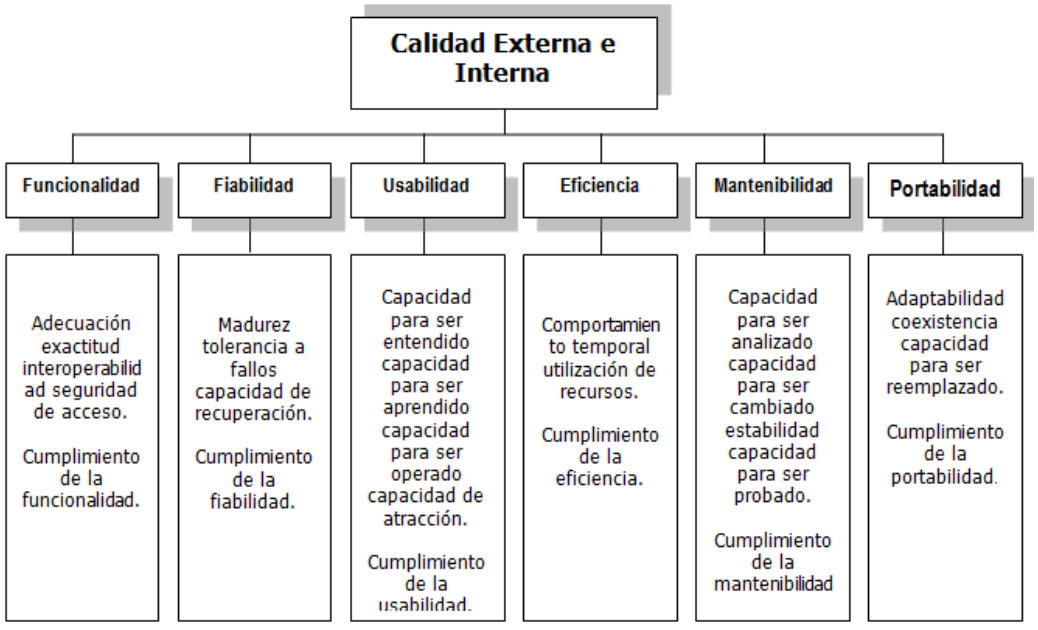


Figura 4.15: Cuadro descriptivo del Estándar 9126.

Haciendo uso de una escala de 1 a 5 puntos, según el siguiente cuadro:

Baremo	Valor
Deficiente	1
Malo	2
Regular	3
Bueno	4
Muy Bueno	5

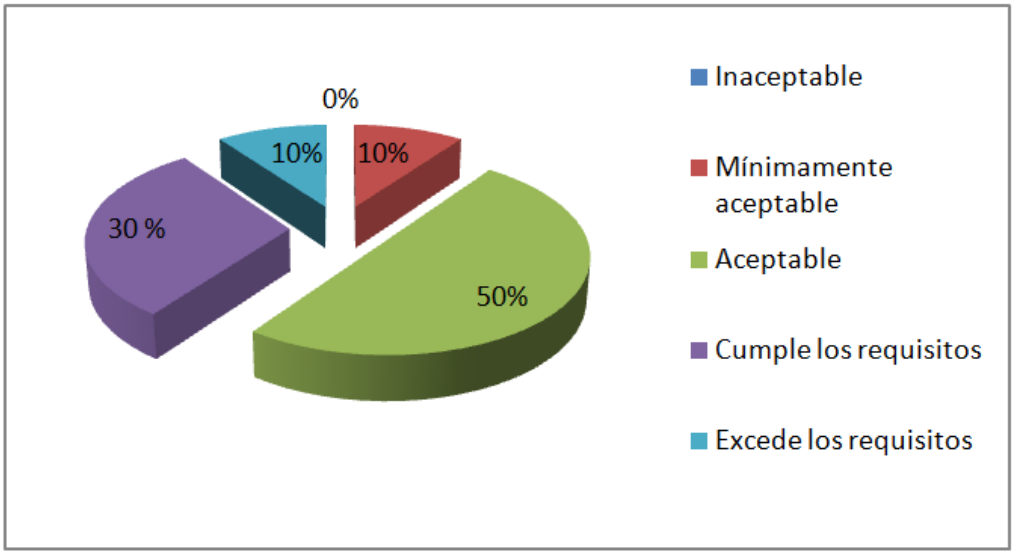
Además se tiene según el estándar, que realizando la sumatoria e los valores que se hayan colocado para cada indicador, se puede tener la siguiente clasificación:

Clasificación	**Intervalo**
Inaceptable	[27 – 54 >
Mínimamente aceptable	[54 – 81 >
Aceptable	[81 – 95 >
Cumple los requisitos	[95 - 122 >
Excede los requisitos	[122 - 135]

Para esta evaluación se considero a **50** alumnos de la Escuela Profesional de Ingeniería Estadística e Informática, los cuales han tenido acceso al Sistema Antivirus y han realizado la clasificación de acuerdo a los indicadores establecidos en la ficha de evaluación.

Los Resultados obtenidos se muestran en el cuadro y gráfico siguiente:

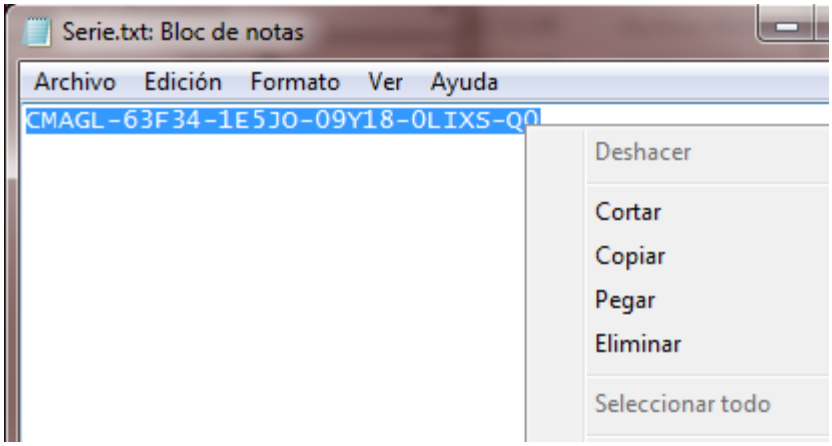
Clasificación	Intervalo	Nº	%
Inaceptable	[27 – 54 >	0	0
Mínimamente aceptable	[54 – 81 >	5	10
Aceptable	[81 – 95 >	25	50
Cumple los requisitos	[95 - 122 >	15	30
Excede los requisitos	[122 - 135]	5	10
Total		50	100



De la totalidad de alumnos o usuarios un 50% de ellos califica como aceptable el desempeño del Software Antivirus **AV-Fenix 2013** y un **30%** le da un puntaje más alto, estableciendo que se brinda un **80%** de protección contra **Malwares** en los dispositivos de almacenamiento extraíbles en los computadores y laptops de los usuarios encuestados.

4.5. INSTALACIÓN

La primera acción será abrir el archivo serie.txt aquí se guarda la serie que permitirá decoficar parte de la información del instalador esto por esquema se seguridad CRC32 y una clave de desencriptado.



Seguidamente cargar el instalador y pegar la clave para finalmente tener instalado el producto, le pedirá que reinicie de forma rápida presione SI y el software se encargara de cerrar la sesión de Windows

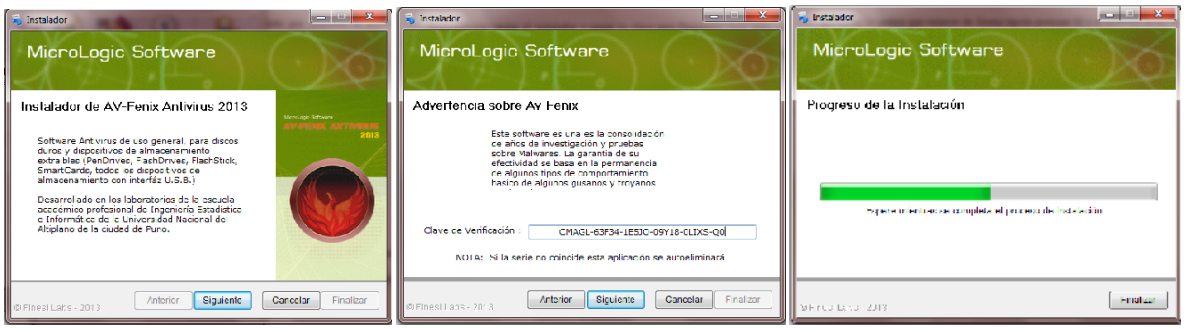


Figura 4.16: Secuencia de Instalación y validación.

4.6. MANUAL DE USUARIO

Todo Software final requerirá el manual de instrucciones para poder contestar preguntas sencillas a los usuarios como:

- **Donde y como conseguir la serie**
- **Instalación recomendada del producto**
- **Posibilidad de descarga y/o envio de comentarios.**
- **Opciones extra y modo de operación**
- **Requerimientos extra.**

Todas estas preguntas se responderán en el manual de Usuario adjunto en esta investigación, **Véase Anexo D**.

CAPÍTULO V

CONCLUSIONES

Primero : Se ha analizado, codificado e implementado el esquemas mas óptimos para desarrollar el Software Antivirus denominado "**AV-Fenix Antivirus 2013"**, el cual en esta versión ofrece protección contras los virus informáticos, que se propagan en los dispositivos de almacenamiento extraíbles (Memorias USB, SD, StickMemory, FlashDrives, etc.).

Segundo : Concluimos que la mejor forma de proteger el computador de las amenazas informáticas latentes en los dispositivos extraíbles, es desarrollando un detector a tiempo real para los dispositivos y notificar al escáner para que este verifique el contenido y pueda informarnos de los resultados, sin tener la necesidad de consultar al usuario pues los virus actúan antes de que el usuario logre responder.

Tercero : Se escribió rutinas de carga y enganche (Hooking) con las APIs de Win32 que permitan notificarnos cuando una instrucción privilegiada esta por ejecutarse, de modo que el detector a tiempo real pueda actuar sin que nosotros intervengamos

ello permite que el programa actué por nosotros así poder protegernos de Malwares.

Cuarto : Se ha analizado, desarrollado e implementado en sitio Web ubicado en la dirección URL http://av-fenix.com, que permitirá a los usuarios del Software Antivirus mantener actualizado una vez subidas las actualizaciones y estas sean descargadas por los usuarios, la actualización contendrá nuevas firmas de virus informáticos, como también la descarga de versiones mejoradas del Software Antivirus, proveyendo una protección general a todos los usuarios de este Software.

CAPÍTULO VI

RECOMENDACIONES

Primero : Se recomienda que el proceso de desarrollo de un software antivirus debe ser constante, el hecho mismo de que los Malware son creados y modificados diariamente hace que los escáner tengan un tiempo de vida limitado, la mejor técnica de detección puede dar la seguridad de protección, pero esta requiere ser mejorada constantemente, así evitar su caducidad.

Segundo : Se recomienda actualizar constantemente la base de datos de ambos lados, en el desarrollo y el uso como usuario regular.

Tercero : Se recomienda investigar otras técnicas de detección y tal vez desarrollar otro producto antivirus igual o mejor, pero siempre teniendo en cuenta que la competencia hace que el producto sea mejor en el tiempo de futuros desarrollos y actualizaciones.

BIBLIOGRAFIA

[A. Borghello, 2001] A. Borghello, C. F. (2001). **Seguridad informática sus implicancias e implementación**.

[Abel, 2002] Abel, Peter. (2002). **El Universo Digital del IBM PC 80386 y Superiores. Anaya Multimedia**, Madrid España, 2da edición,. Reimpreso en Abril de 1988.

[Tanenbaum, 2006] Andrew S. Tanenbaum, A. S. W. (January 04, 2006). **Operating Systems Design and Implementation**. Prentice Hall, third edition.

[Burns, 2003] Burns, K. (2003). **TCP/IP Analysis and Troubleshooting Toolkit. Computer network protocol**. Wiley, Hoboken, NJ, USA, 1st edition.

[Chromatic, 2004] Chromatic (2004). **Guía de Bolsillo de Programación Extrema**, p.10. O'Reilly, 1th edición.

[de Celis, 1997] de Celis, C. G. (1997). **El Universo Digital del IBM PC AT y PS/2**. Grupo Informático 1992-1997. Anaya Multimedia, Casa del Estudiante Av. Real Burgos S/N, 4ta edición.

[Eco, 2005] Eco, U. (2005). **Como se hace una Tesis, Técnicas y procedimientos de estudio, investigación y escritura**. Universidad de Sevilla, 1st edición.

[Juares, 2004] Juares Henry. (2004). **Protección de programas ejecutables usando autómatas celulares de Wolfgang.**, Tesis de grado, Facultad de Ingeniería Estadística e Informática, UNA Puno.

[Kenneth, 2001] Kenneth Calvert, M. D. (10/2001). **TCP/IP Sockets in Java - Practical Guide for Programmers**. Morgan Kaufmann, Burlington, MA, USA, 1st edition.

[Knuth, 1984] Knuth, D. E. (1984). **The TEX Book**. **Computers & Typesetting**. Addison-Wesley, Reading, Massachusetts, 15th edition. Reprinted as Vol. A of Computers & Typesetting, 1986.

[Rogers, 2003] Rogers, D. T. (Junio de 2003). **Un Framework para la Subversión Dinámica. Glosario de Multics acrónimos y términos**. 1st edition. Seguridad y Anillos.

[Rosello, 1988] Rosello, M. A. R. (1988). **8088-8086/8087 Programación Ensamblador en entorno MS-DOS**. Anaya Multimedia, Madrid España, 2da edición. Reimpreso en Abril de 1988.

[Smart, 2006] Julian Smart, K. H. , (2006). **Cross-Platform GUI Programming with wx-Widgets**. Prentice Hall PTR, Copyright c 2006 Pearson Education, Inc., 1st edition. ISBN 0-13-147381-6.

[Walker, 2006] Walker, A. (2006). **Seguridad, Spam, Spyware y Virus. Seguridad Informatics**. Anaya Multimedia-Anaya Interactiva, Reading, Massachusetts, 1st edition.(Julio, 2006).

ANEXO A

CODIGO FUENTE AV-FENIX VERSION WIN32

```
01:
02: // AvFenix.cpp : the application.
03:
04: #include "stdafx.h"
05: #include "AvFenix.h"
06:
07: #include "MainFrm.h"
08: #include "AvFenixDoc.h"
09: #include "AvFenixView.h"
10:
11: #include "AvFenixAbout.h"
12:
13:
14: BEGIN_MESSAGE_MAP(CAvFenixApp, CWinApp)
15:     //{{AFX_MSG_MAP(CAvFenixApp)
16:     ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
17:     //}}AFX_MSG_MAP
18: END_MESSAGE_MAP()
19:
20:
21: CAvFenixApp::CAvFenixApp()
22: {
23:     // Load Libs and Others
24: }
25:
26: CAvFenixApp theApp;
27:
28:
29: BOOL CAvFenixApp::InitInstance()
30: {
31:     // let's go begin
32:     AvFnxStart();
33:
34:
35:     // aqui el resto de sentencias de creacion visual
36:     AfxEnableControlContainer();
37:     SetRegistryKey(_T("Local AppWizard-Generated Applications"));
38:     LoadStdProfileSettings(0);
39:     InitCommonControls();
40:     CSingleDocTemplate* pDocTemplate;
41:     pDocTemplate = new CSingleDocTemplate(
42:         IDR_MAINFRAME,
43:         RUNTIME_CLASS(CAvFenixDoc),
44:         RUNTIME_CLASS(CMainFrame),
45:         RUNTIME_CLASS(CAvFenixView));
46:     AddDocTemplate(pDocTemplate);
47:
48:     CCommandLineInfo cmdInfo;
```

```
49:     ParseCommandLine(cmdInfo);
50:
51:     if (!ProcessShellCommand(cmdInfo))
52:         return FALSE;
53:
54:     //m_pMainWnd->SetWindowPos( NULL, 0, 0, 600, 500, SWP_NOMOVE );
55:     m_pMainWnd->SetWindowText( "AV-Fenix Antivirus 2013" );
56:
57:     //m_pMainWnd->ShowWindow(SW_SHOWMINIMIZED);
58:     m_pMainWnd->UpdateWindow();
59:
60:     return TRUE;
61: }
62:
63:
64:
65: // en : #include "AvFenixAbout.h"
66:
67: CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
68: {
69:     //{{AFX_DATA_INIT(CAboutDlg)
70:     //}}AFX_DATA_INIT
71: }
72:
73: void CAboutDlg::DoDataExchange(CDataExchange* pDX)
74: {
75:     CDialog::DoDataExchange(pDX);
76:     //{{AFX_DATA_MAP(CAboutDlg)
77:     //}}AFX_DATA_MAP
78: }
79:
80: BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
81:     //{{AFX_MSG_MAP(CAboutDlg)
82:     //}}AFX_MSG_MAP
83: END_MESSAGE_MAP()
84:
85:
86:
87: void CAvFenixApp::OnAppAbout()
88: {
89:     CAboutDlg aboutDlg;
90:     aboutDlg.DoModal();
91: }
92:
93:
94: BOOL CAboutDlg::OnInitDialog()
95: {
96:     CDialog::OnInitDialog();
97:     mlSetAlpha( m_hWnd, 225 );
98:     return TRUE;
99: }
```

```
01:
02: // AvFenix.h : main header
03:
04: #if !defined (AFX_AVFENIX_H_)
05: #define AFX_AVFENIX_H_
06:
07:
08: #ifndef __AFXWIN_H__
09:     #error include 'stdafx.h' before including this file for PCH
10: #endif
11:
12: #include "resource.h"
13:
14:
15: class CAvFenixApp : public CWinApp
16: {
17:   public:
18:     CAvFenixApp();
19:     //{{AFX_VIRTUAL(CAvFenixApp)
20:     public:
21:     virtual BOOL InitInstance();
22:     //}}AFX_VIRTUAL
23:
24:     //{{AFX_MSG(CAvFenixApp)
25:     afx_msg void OnAppAbout();
26:     //}}AFX_MSG
27:     DECLARE_MESSAGE_MAP()
28: };
29:
30: #endif
31:
```

```
01:
02: // MainFrm.cpp : implementation of the CMainFrame class
03:
04: #include "stdafx.h"
05: #include "AvFenix.h"
06: #include "MainFrm.h"
07:
08: IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
09: BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
10:     //{{AFX_MSG_MAP(CMainFrame)
11:     ON_COMMAND(MNU_PROG_CERRAR, OnProgCerrar)
12:     ON_COMMAND(MNU_PROG_MINIM, OnProgMinim)
13:     ON_WM_CLOSE()
14:     ON_WM_ACTIVATEAPP()
15:     ON_WM_TIMER()
16:     ON_WM_ACTIVATE()
17:     ON_WM_CREATE()
18:     ON_WM_PAINT()
19:     //}}AFX_MSG_MAP
20: END_MESSAGE_MAP()
21:
22:
23: CMainFrame::CMainFrame()
24: {
25:
26: }
27:
28: CMainFrame::~CMainFrame()
29: {
```

```
30:
31: }
32:
33: //-------------------------------------------------------------
34: // Eventos administrados
35: //-------------------------------------------------------------
36: #define FnxEv_Start  1
37: #define FnxEv_Pause  2
38: #define FnxEv_Stop   3
39: #define FnxEv_Mnu1   10
40: #define FnxEv_Mnu2   11
41: #define FnxEv_Mnu3   12
42:
43: //-------------------------------------------------------------
44: // Rererencia hacia nuestro Administrador
45: //-------------------------------------------------------------
46: CMainFrame *ptrControl;
47:
48: void AvFnxHandlerEvents( int ev )
49: {
50:     switch( ev )
51:     {
52:       case FnxEv_Start:
53:           // Contacto iniciado
54:           break;
55:
56:       case FnxEv_Mnu1:
57:           // Centro de control
58:           ptrControl->ShowWindow( SW_SHOWNORMAL );
59:           break;
60:     }
61: }
62:
63: BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
64: {
65:     if( !CFrameWnd::PreCreateWindow(cs) )
66:         return FALSE;
67:
68:
69:     cs.style = WS_OVERLAPPED | WS_CAPTION | FWS_ADDTOTITLE
70:         | WS_DLGFRAME | WS_SYSMENU | WS_MINIMIZEBOX;
71:         // | WS_THICKFRAME | WS_SYSMENU | WS_MINIMIZEBOX;
72:
73:     cs.cx = 710;
74:     cs.cy = 524;
75:
76:     // Put the Hook
77:     AvFnxSetHook( AvFnxHandlerEvents );
78:     ptrControl = this;
79:
80:
81:     return TRUE;
82: }
83:
84: void CMainFrame::OnProgCerrar()
85: {
86:     if( mlShowMessage("Porfavor confirme si realmente desea cerrar el
                          Antivirus", "Cerrar AV-Fenix", MB_YESNO ) == IDOK )
87:     {
88:         AvFnxStop();
89:         this->CloseWindow();
90:         this->DestroyWindow();
91:         //CFrameWnd::OnClose();
```

```
92:     }
93: }
94:
95: void CMainFrame::OnProgMinim()
96: {
97:     // minimizar y esconder
98:     this->CloseWindow();
99:     this->ShowWindow( SW_HIDE );
100: }
101:
102: void CMainFrame::OnClose()
103: {
104:     // llamar eventos
105:     OnProgMinim();
106:     // OnClose();
107: }
108:
109: void CMainFrame::OnActivateApp(BOOL bActive, HTASK hTask)
110: {
111:     CFrameWnd::OnActivateApp(bActive, hTask);
112:     mlSetAlpha( m_hWnd, 250 );
113: }
114:
115: void CMainFrame::OnTimer(UINT nIDEvent)
116: {
117:
118:     CFrameWnd::OnTimer(nIDEvent);
119: }
120:
```

```
01:
02: // AvFenixView.cpp : implementation of the CAvFenixView class
03:
04: #include "stdafx.h"
05: #include "AvFenix.h"
06:
07: #include "AvFenixDoc.h"
08: #include "AvFenixView.h"
09:
10: #include "AvFenixAbout.h"
11:
12: IMPLEMENT_DYNCREATE(CAvFenixView, CView)
13:
14: BEGIN_MESSAGE_MAP(CAvFenixView, CView)
15:     //{{AFX_MSG_MAP(CAvFenixView)
16:     ON_WM_MOUSEMOVE()
17:     ON_WM_LBUTTONUP()
18:     ON_WM_ERASEBKGND()
19:     //}}AFX_MSG_MAP
20: END_MESSAGE_MAP()
21:
22: CAvFenixView::CAvFenixView()
23: {
24:     nCursor = -1;
25:     nPrevCur = -2;
26: }
27:
28: CAvFenixView::~CAvFenixView()
29: {
30:
31: }
```

```
32:
33: BOOL CAvFenixView::PreCreateWindow(CREATESTRUCT& cs)
34: {
35:     return CView::PreCreateWindow(cs);
36: }
37:
38: BOOL CAvFenixView::OnEraseBkgnd(CDC* pDC)
39: {
40:     //return CView::OnEraseBkgnd(pDC);
41:     return FALSE;
42: }
43:
44: int ArrMenuPos[]= {
45:     130, 130+60,
46:     195, 195+60,
47:     259, 259+60,
48:     324, 324+60
49: };
50:
51:
52: char *lpTips[] = {
53:     "Busqueda por demanda, iniciará un dialogo para personalizar busquedas y
        detección de Malwares,",
54:     "Actualizar base de firmas de Malwares. Tambien puede descargarse
        manualmente desde http://www.av-fenix.com",
55:     "Información técnica sobre el desarrollo de este Software.",
56:     "Cerrar antivirus. Precaución al cerrar este Software Antivirus su PC
        quedará desprotegida."
57: };
58:
59:
60: void CAvFenixView::OnDraw(CDC* pDC)
61: {
62:     CRect mRect;
63:     GetClientRect( mRect );
64:
65:     CBitmap        hBmp;
66:     hBmp.LoadBitmap( BMP_MENU );
67:
68:     CDC            hBk;
69:     hBk.CreateCompatibleDC( NULL );
70:     hBk.SelectObject( &hBmp );
71:
72:     LOGFONT hLogFont;
73:     CFont   hFont;
74:     CFont  *pFtOld;
75:
76:
77:     if( nCursor>=0 && nCursor<4 )
78:     {
79:         hBk.SelectStockObject( NULL_BRUSH );
80:         hBk.Rectangle( CRect(  71, ArrMenuPos[nCursor *2],
81:                               321, ArrMenuPos[nCursor *2+1]) );
82:
83:         memset(&hLogFont, 0, sizeof(LOGFONT));
84:         hLogFont.lfHeight = 22;
85:         //hLogFont.lfWeight = FW_BOLD;
86:         lstrcpy(hLogFont.lfFaceName, "Verdana");
87:         hFont.CreateFontIndirect(&hLogFont);
88:         pFtOld = hBk.SelectObject(&hFont);
89:
90:         mRect = CRect(370,170,600,300);
91:         //hBk.Rectangle( mRect );
```

```
92:         hBk.SetBkMode( TRANSPARENT );
93:         hBk.SetTextColor( RGB(80,120,80) );
94:         hBk.DrawText( CString(lpTips[nCursor]), mRect, DT_CENTER |
                                                DT_VCENTER | DT_WORDBREAK);
95:
96:         // Restaurar los componentes de CDC
97:         pDC->SelectObject(pFtOld);
98:     }
99:     //-----------------------------------------
100:     pDC->BitBlt(0,0,702,472, &hBk,0,0, SRCCOPY );
101:
102:     hBk.DeleteDC();
103:     hBmp.DeleteObject();
104:
105: }
106:
107: void CAvFenixView::OnMouseMove(UINT nFlags, CPoint point)
108: {
109:     nCursor = -1;
110:     for( int i=0; i<4; i++ )
111:     {
112:         if( point.x > 71 && point.x < 321 )
113:         if( point.y > ArrMenuPos[i*2] && point.y < ArrMenuPos[i*2+1] )
114:             nCursor = i;
115:     }
116:
117:     // prevenir Blink
118:     if( nCursor != nPrevCur )
119:     {
120:         MessageBeep( 1870 );
121:         nPrevCur = nCursor;
122:         Invalidate();
123:     }
124:
125:     CView::OnMouseMove(nFlags, point);
126: }
127:
128: void CAvFenixView::OnLButtonUp(UINT nFlags, CPoint point)
129: {
130:     // seleccionar
131:
132:     if( nCursor == 0 )
133:     {
134:         AvFnxControl();
135:     }
136:
137:     if( nCursor == 1 )
138:     {
139:         AvFnxUpdate();
140:     }
141:
142:     if( nCursor == 2 )
143:     {
144:         CAboutDlg aboutDlg;
145:         aboutDlg.DoModal();
146:     }
147:
148:     if( nCursor == 3 )
149:     {
150:         if( mlShowMessage("Porfavor confirme si realmente desea cerrar el
                Antivirus", "Cerrar AV-Fenix", MB_YESNO ) == IDOK )
151:         {
152:             AvFnxStop();
```

```
153:             GetParent()->CloseWindow();
154:             GetParent()->DestroyWindow();
155:         }
156:     }
157:
158:
159:     CView::OnLButtonUp(nFlags, point);
160: }
161:
162:
```

```
01:
02: #if !defined (AFX_STDAFX_H_)
03: #define AFX_STDAFX_H_
04:
05: #if _MSC_VER > 1000
06: #pragma once
07: #endif // _MSC_VER > 1000
08:
09: #define VC_EXTRALEAN
10:
11: #include <afxwin.h>
12: #include <afxext.h>
13: #include <afxdisp.h>
14: #include <afxdtctl.h>
15: #ifndef _AFX_NO_AFXCMN_SUPPORT
16: #include <afxcmn.h>
17: #endif
18:
19:
20: #ifdef _DEBUG
21: #define new DEBUG_NEW
22: #undef THIS_FILE
23: static char THIS_FILE[] = __FILE__;
24: #endif
25:
26:
27:
28:
29: //---------------------------------------------------------------------
30: // Area de Importacion de funciones
31: //---------------------------------------------------------------------
32: extern "C" void AvFnxStart();
33: extern "C" void AvFnxStop();
34: extern "C" void AvFnxScanDir( char *lpPath );
35: extern "C" void AvFnxSetHook( void (*lpEvents)(int ev) );
36: extern "C" void AvFnxControl();
37: extern "C" void AvFnxUpdate();
38: extern "C" UINT mlShowMessage( LPSTR, LPSTR, UINT );
39: extern "C" void mlSetAlpha( HWND, UCHAR  );
40: //---------------------------------------------------------------------
41:
42: #endif
43:
```

ANEXO B

CODIGO FUENTE AV-FENIX wxWidgets Win/Linux

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<CodeBlocks_project_file>
	<FileVersion major="1" minor="6" />
	<Project>
		<Option title="AvFenixApp" />
		<Option pch_mode="2" />
		<Option compiler="gcc" />
		<Build>
			<Target title="Debug">
				<Option output="bin\Debug\AvFenixApp" prefix_auto="1"
                                        extension_auto="1" />
				<Option object_output="obj\Debug\" />
				<Option type="0" />
				<Option compiler="gcc" />
				<Option projectLinkerOptionsRelation="2" />
				<Compiler>
					<Add option="-g" />
					<Add option="-D__WXDEBUG__" />
					<Add directory="C:\ML-GnuSDK\wxWidgets-
                                                2.8.7\lib\gcc_lib\mswd" />
				</Compiler>
				<ResourceCompiler>
					<Add directory="C:\ML-GnuSDK\wxWidgets-
                                                2.8.7\lib\gcc_lib\mswd" />
				</ResourceCompiler>
				<Linker>
					<Add library="libwxmsw28d.a" />
					<Add library="libwxpngd.a" />
					<Add library="libwxjpegd.a" />
					<Add library="libwxtiffd.a" />
					<Add library="libwxzlibd.a" />
					<Add directory="C:\ML-GnuSDK\wxWidgets-
                                           2.8.7\lib\gcc_lib" />
				</Linker>
			</Target>
			<Target title="Release">
				<Option output="bin\Release\AvFenixApp" prefix_auto="1"
                                         extension_auto="1" />
				<Option object_output="obj\Release\" />
				<Option type="0" />
				<Option compiler="gcc" />
				<Option projectLinkerOptionsRelation="2" />
				<Compiler>
					<Add option="-O2" />
					<Add directory="C:\ML-GnuSDK\wxWidgets-
                                         2.8.7\lib\gcc_lib\msw" />
				</Compiler>
				<ResourceCompiler>
					<Add directory="C:\ML-GnuSDK\wxWidgets-
                                         2.8.7\lib\gcc_lib\msw" />
```

```
                        </ResourceCompiler>
                        <Linker>
                              <Add option="-s" />
                              <Add library="libwxmsw28.a" />
                              <Add library="libwxpng.a" />
                              <Add library="libwxjpeg.a" />
                              <Add library="libwxtiff.a" />
                              <Add library="libwxzlib.a" />
                              <Add directory="C:\ML-GnuSDK\wxWidgets-
                                    2.8.7\lib\gcc_lib" />
                        </Linker>
                  </Target>
            </Build>
            <Compiler>
                  <Add option="-pipe" />
                  <Add option="-mthreads" />
                  <Add option="-D__GNUWIN32__" />
                  <Add option="-D__WXMSW__" />
                  <Add option='[[if (PLATFORM == PLATFORM_MSW &amp;amp;&amp;
                   (GetCompilerFactory().GetCompilerVersionString( )
                  <Add option="-Wall" />
                  <Add directory="C:\ML-GnuSDK\wxWidgets-2.8.7\include" />
                  <Add directory="C:\ML-GnuSDK\wxWidgets-2.8.7\contrib\include"
              />
            </Compiler>
            <ResourceCompiler>
                  <Add directory="C:\ML-GnuSDK\wxWidgets-2.8.7\include" />
            </ResourceCompiler>
            <Linker>
                  <Add library="libkernel32.a" />
                  <Add library="libuser32.a" />
                  <Add library="libgdi32.a" />
                  <Add library="libwinspool.a" />
                  <Add library="libcomdlg32.a" />
                  <Add library="libadvapi32.a" />
                  <Add library="libshell32.a" />
                  <Add library="libole32.a" />
                  <Add library="liboleaut32.a" />
                  <Add library="libuuid.a" />
                  <Add library="libcomctl32.a" />
                  <Add library="libwsock32.a" />
                  <Add library="libodbc32.a" />
            </Linker>
            <Unit filename="FenixApp.cpp" />
            <Unit filename="FenixCnvs.cpp" />
            <Unit filename="FenixCnvs.h" />
            <Unit filename="FenixFrm.cpp" />
            <Unit filename="FenixFrm.h" />
            <Unit filename="Global.h" />
            <Extensions>
                  <code_completion />
                  <envvars />
                  <debugger />
            </Extensions>
      </Project>
</CodeBlocks_project_file>
```

```
/*****************************************************************
 *
 * Project Name : AvFenix Antivirus  [kronoz-av]
 * Version      : 2.0.a
 *
 * Developer    : Ramiro Pedro Laura Murillo
 *              : Developed as Thesis Degree
 *
 * Date & Time  : January of 2012
 *
 *****************************************************************/



#include "Global.h"
#include "FenixFrm.h"
#include "FenixCnvs.h"


class AvFenixApp : public wxApp
{
    public:
        AvFenixApp(void){ };
        bool OnInit();
};


bool AvFenixApp::OnInit(void)
{
    // we add to PNG supoort
    //
    wxImage::AddHandler( new wxPNGHandler );


    AvFenixFrm *FnxFrame = new AvFenixFrm( NULL );

    AvFenixCnvs *canvas = new AvFenixCnvs( FnxFrame, wxPoint(0, 0),
                                wxSize(100, 100) );
    FnxFrame->pCanvas = canvas;

    FnxFrame->SetTitle( _("Av-Fenix Antivirus - 2012") );
    FnxFrame->Show(true);

    return true;
}

IMPLEMENT_APP(AvFenixApp)
```

```
/*****************************************************************
 *
 * Project Name : AvFenix Antivirus  [kronoz-av]
 * Version      : 2.0.a
 *
 * Developer    : Ramiro Pedro Laura Murillo
 *              : Developed as Thesis Degree
 *
 * Date & Time  : January of 2012
 *
 *****************************************************************/


#include "FenixCnvs.h"

#include <wx/intl.h>
#include <wx/string.h>


BEGIN_EVENT_TABLE(AvFenixCnvs,wxScrolledWindow)
       //(*EventTable(AvFenixCnvs)
       //*)
END_EVENT_TABLE()


AvFenixCnvs::AvFenixCnvs(wxWindow *parent, const wxPoint& pos, const wxSize& size)
:
                          wxScrolledWindow(parent, wxID_ANY, pos, size )
{

       SetClientSize(wxDefaultSize);
       Move(wxDefaultPosition);

       Connect(wxEVT_PAINT,(wxObjectEventFunction)&AvFenixCnvs::OnPaint);

    imgBase = (wxBitmap *) NULL;
    imgBox1 = (wxBitmap *) NULL;
    imgBox1 = (wxBitmap *) NULL;
    imgBox1 = (wxBitmap *) NULL;
    imgBox1 = (wxBitmap *) NULL;
    imgBox1 = (wxBitmap *) NULL;

    imgBase = LoadBitmapFromPng( _("res/FnxBase.png") );
    imgBox1 = LoadBitmapFromPng( _("res/FnxBox1.png") );
    //imgBox2 = LoadBitmapFromPng( _("res/FnxBox2.png") );
    //imgLogo = LoadBitmapFromPng( _("res/FnxLogo.png") );

    imgMnu1 = LoadBitmapFromPng( _("res/Menu1.png") );
    imgMnu2 = LoadBitmapFromPng( _("res/Menu2.png") );
}

wxBitmap *AvFenixCnvs::LoadBitmapFromPng( wxString pngFile )
{
    wxBitmap *tmpBitmap = new wxBitmap( pngFile, wxBITMAP_TYPE_PNG );
    if( !tmpBitmap->Ok() )
    {
        delete imgBase;
        return (wxBitmap*) NULL;
    }

    return tmpBitmap;
}
```

```
AvFenixCnvs::~AvFenixCnvs()
{

}


void AvFenixCnvs::OnPaint(wxPaintEvent& event)
{
    wxPaintDC dc(this);

    /*
    if( imgBase->Ok() )
    {
        wxMemoryDC memDC;
        memDC.SelectObject(*imgBase);

        // Transparent blitting if there's a mask in the bitmap
        dc.Blit( 0 , 0, imgBase->GetWidth(), imgBase->GetHeight(), & memDC,
            0, 0, wxCOPY, true);

        memDC.SelectObject(wxNullBitmap);
    }
    */

    DrawImage( &dc, 0, 0, imgBase );
    //DrawImage( &dc, 320, 7, imgLogo );

    DrawImage( &dc, 40, 100, imgBox1 );
    //DrawImage( &dc, 40, 457, imgBox2 );

    DrawImage( &dc, 70, 130+0 , imgMnu1 );
    DrawImage( &dc, 70, 130+64, imgMnu2 );

}

void AvFenixCnvs::DrawImage( wxPaintDC *pDC, int x, int y, wxBitmap *tmpImage )
{
    if( tmpImage->Ok() )
    {
        wxMemoryDC memDC;
        memDC.SelectObject(*tmpImage);

        // Transparent blitting if there's a mask in the bitmap
        pDC->Blit( x , y, tmpImage->GetWidth(), tmpImage->GetHeight(),
                &memDC, 0, 0, wxCOPY, true);

        memDC.SelectObject(wxNullBitmap);
    }
}
```

```
/*************************************************************************
 *
 * Project Name : AvFenix Antivirus  [kronoz-av]
 * Version      : 2.0.a
 *
 * Developer    : Ramiro Pedro Laura Murillo
 *              : Developed as Thesis Degree
 *
 * Date & Time  : January of 2012
 *
 ************************************************************************/


#include "FenixFrm.h"


//(*InternalHeaders(AvFenixFrm)
#include <wx/intl.h>
#include <wx/string.h>
//*)

#include "FenixCnvs.h"


//(*IdInit(AvFenixFrm)
//*)

BEGIN_EVENT_TABLE(AvFenixFrm,wxFrame)
       //(*EventTable(AvFenixFrm)
       //*)
END_EVENT_TABLE()


AvFenixFrm::AvFenixFrm( wxWindow* parent,wxWindowID id,
                        const wxPoint& pos,const wxSize& size)
{
    pCanvas = (AvFenixCnvs *) NULL;

       //(*Initialize(AvFenixFrm)
       Create(parent, id, wxEmptyString, wxDefaultPosition,
                          wxDefaultSize, wxCAPTION | wxSYSTEM_MENU |
                               wxCLOSE_BOX | wxMINIMIZE_BOX, _T("id"));

       SetClientSize(wxSize(700,470));
       Move(wxDefaultPosition);
       Center();
       //*)
}

AvFenixFrm::~AvFenixFrm()
{
       //(*Destroy(AvFenixFrm)
       //*)
}
```

```
/*****************************************************************
 *
 * Project Name : AvFenix Antivirus  [kronoz-av]
 * Version      : 2.0.a
 *
 * Developer    : Ramiro Pedro Laura Murillo
 *              : Developed as Thesis Degree
 *
 * Date & Time  : January of 2012
 *
 *****************************************************************/

#ifndef FENIXCNVS_H
#define FENIXCNVS_H

#include "Global.h"


class AvFenixCnvs: public wxScrolledWindow
{
       public:
         AvFenixCnvs(wxWindow *parent, const wxPoint& pos, const wxSize& size);
              virtual ~AvFenixCnvs();

       protected:
         wxBitmap  *imgBase;
         wxBitmap  *imgBox1;
         wxBitmap  *imgBox2;
         wxBitmap  *imgMnu1;
         wxBitmap  *imgMnu2;
         wxBitmap  *imgLogo;

         wxBitmap *LoadBitmapFromPng( wxString pngFile );
         void DrawImage( wxPaintDC *pDC, int x, int y, wxBitmap *tmpImage );

       private:

              //(*Handlers(AvFenixCnvs)
              void OnPaint(wxPaintEvent& event);
              //*)

              DECLARE_EVENT_TABLE()
};

#endif
```

```
/*************************************************************************
 *
 * Project Name : AvFenix Antivirus  [kronoz-av]
 * Version      : 2.0.a
 *
 * Developer    : Ramiro Pedro Laura Murillo
 *              : Developed as Thesis Degree
 *
 * Date & Time  : January of 2012
 *
 *************************************************************************/

#ifndef FENIXFRM_H
#define FENIXFRM_H

//(*Headers(AvFenixFrm)
#include <wx/dialog.h>
//*)

#include "Global.h"

class AvFenixCnvs;

class AvFenixFrm: public wxFrame
{
       public:
         AvFenixCnvs *pCanvas;

              AvFenixFrm( wxWindow* parent, wxWindowID id=wxID_ANY,
                     const wxPoint &pos=wxDefaultPosition,
                     const wxSize  &size=wxDefaultSize );

              virtual ~AvFenixFrm();

              DECLARE_EVENT_TABLE()
};

#endif
```

```
// archive : Global.h
// para importacion de DLL y APIs extra
//

#include <wx/wx.h>
#include <wx/image.h>

#include "wx/wxprec.h"
```

ANEXO C

CODIGO FUENTE AV-FENIX EN ANDROID SDK

Archivo: AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.avfenix"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Archivo: .project

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
       <name>AvFenix</name>
       <comment></comment>
       <projects>
       </projects>
       <buildSpec>
              <buildCommand>

       <name>com.android.ide.eclipse.adt.ResourceManagerBuilder</name>
                     <arguments>
                     </arguments>
              </buildCommand>
              <buildCommand>
                     <name>com.android.ide.eclipse.adt.PreCompilerBuilder</name>
```

```
                    <arguments>
                    </arguments>
             </buildCommand>
             <buildCommand>
                    <name>org.eclipse.jdt.core.javabuilder</name>
                    <arguments>
                    </arguments>
             </buildCommand>
             <buildCommand>
                    <name>com.android.ide.eclipse.adt.ApkBuilder</name>
                    <arguments>
                    </arguments>
             </buildCommand>
      </buildSpec>
      <natures>
             <nature>com.android.ide.eclipse.adt.AndroidNature</nature>
             <nature>org.eclipse.jdt.core.javanature</nature>
      </natures>
      <linkedResources>
             <link>
                    <name>res/drawable-hdpi/droidfenix.png</name>
                    <type>1</type>
                    <location>/home/kronoz/workspace/AvFenix/res/drawable-
                           hdpi/droidfenix.png</location>
             </link>
             <link>
                    <name>res/drawable-hdpi/fnx_logo.png</name>
                    <type>1</type>
                    <location>/home/kronoz/workspace/AvFenix/res/drawable-
                           hdpi/fnx_logo.png</location>
             </link>
             <link>
                    <name>res/drawable-hdpi/ic_search.png</name>
                    <type>1</type>
                    <location>/home/kronoz/workspace/AvFenix/res/drawable-
                           hdpi/ic_search.png</location>
             </link>
      </linkedResources>
</projectDescription>
```

Archivo: .classpath

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
      <classpathentry kind="src" path="src"/>
      <classpathentry kind="src" path="gen"/>
      <classpathentry kind="con" path=
                    "com.android.ide.eclipse.adt.ANDROID_FRAMEWORK"/>
      <classpathentry kind="con" path="com.android.ide.eclipse.adt.LIBRARIES"/>
      <classpathentry kind="output" path="bin/classes"/>
</classpath>
```

Archivo: activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="182dp"
        android:orientation="vertical" >
    </LinearLayout>

    <View
        android:id="@+id/view1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:background="@drawable/droidfenix" />

</RelativeLayout>
```

Archivo: MainActiviry.java

```
/*****************************************************************
 *
 * Project Name : AvFenix Antivirus  2013 Android
 * Version      : 2.0.a
 *
 * Developer    : Ramiro Pedro Laura Murillo
 *              : Developed as Thesis Degree
 *
 * Date & Time  : January of 2013
 *
 *****************************************************************/



package com.example.avfenix;



import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.view.Menu;
import android.view.MenuItem;
```

```
public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // getMenuInflater().inflate(R.menu.activity_main, menu);

        super.onCreateOptionsMenu(menu);
        menu.add(0, 1, 0, "Escanear")
              .setIcon(R.drawable.ic_search );
        menu.add(0, 2, 0, "About")
              .setIcon(R.drawable.ic_search );
        menu.add(0, 3, 0, "Salir")
              .setIcon(R.drawable.ic_search );

        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

       //int t = item.getItemId();
       //setTitle( "=" + t );

        switch ( item.getItemId() ) {
              case 1 :
                     cmdScan();
              return true;

              case 2 :
                     cmdAbout();
                     return true;

              case 3:
                     cmdExit();
                     return true;

              default:
                     return super.onOptionsItemSelected(item);
        }
    }

    public void cmdScan()
    {
       AlertDialog.Builder builder = new AlertDialog.Builder(this);
       builder.setMessage("Direccion: \\media\\sd \nFinalizado !")
                .setTitle("Escaneo por demanda")
                .setCancelable(false)
                .setNeutralButton("Cerrar",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });
```

```
        //AlertDialog alert = builder.create();
        builder.create().show();
        //alert.show();
    }

    public void cmdAbout()
    {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("AV-Fenix Antivirus 2013 version beta
                            para Andorid OS Desarrollado en (c)FinesiLabs")
                .setTitle("Info")
                .setCancelable(false)
                .setNeutralButton("Cerrar",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });

        builder.create().show();
    }

    public void cmdExit()
    {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);

        builder.setMessage("¿En verdad desea salir?")
                .setTitle("Advertencia")
                .setCancelable(false)
                .setNegativeButton("Cancelar",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        })
                .setPositiveButton("Continuar",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                   MainActivity.this.finish();
                            }
                        });

        builder.create().show();
    }
}
```

ANEXO D

CODIGO FUENTE AV-FENIX KrystalView


```
01:
02: /**************************************************************************
03:  * Kronoz Project - Av-Fenix Antivirus
04:  * Modulo : KrystalView Environment + [KronozVideo Agent]
05:  * Fecha  : Febrero de 2012
06:  *
07:  * Historial :
08:  *
09:  *   Enero   15 : Se revisa KronozVesa de 32K y 64K realiza mejoras
10:  *   Febrero 01 : Iniciamos VesaAgent y para 32BPP 24M de color
11:  *                usa char[4] para acceder a VGA
12:  *   Febrero 15 : Mejora con ASM de 32bits para almacenar STOSD
13:  *                cambio de compilador fallido y mejora del proceso
14:  *                a) BCC -ml /B /3 kvDemo.cpp  ASM->386->exe   [ojo]
15:  *   Febrero 16 : Mejoras con DWORD kMemSet para mayor velocidad de
16:  *                dibujo y mejora de las funciones Blend y Bitmap
17:  *                Bankos controlados y "DWORD kVGA[]" para acceso
18:  *                (problemas con union struct conflictos y mucho ASM
19:  *                generando exceso de codigo, optimizado a mano..)
20:  *
21:  *
22:  *      Byte  | B | G | R | A |
23:  *            +---------------+
24:  *            | 0 | 1 | 2 | 3 |
25:  *
26:  *     ((Color.Blue<<8 | Color.Red)<<8 | Color.Blue)>>8 | Alpha
27:  *
28:  *     DWORD | A | R | G | B |  -- intel almacena como |B|G|R|A|
29:  *
30:  *
31:  *
32:  *   Febrero 17 : Optimizacion y generacion de KrystalView Demos
33:  *                Simplificaion y optimizacion de COLORREF - RGB
34:  *
35:  **************************************************************************/
```

```
36:
37:
38: #if !defined _KrystalViewGraph_H
39: #define _KrystalViewGraph_H
40:
41: #include <stdio.h>
42: #include <conio.h>
43: #include <math.h>
44: #include <dos.h>
45:
46: #include "kvDefs.h"
47: #include "kvMem.h"
48:
49:
50: #define KV_VESA32  2
51: #define KV_VGA4    1
52: #define KV_C8025   0
53:
54:
55: // VGA Array DMA Buffer on  Real Mode and ProtectMode
56: // A000:0000
57: //
58: //unsigned char far *kVGA = (unsigned char far*) 0xA0000000L;  // 8bits
59: unsigned long  far *kVGA = (unsigned long far*) 0xA0000000L;  // 32bits
A000:0000
60:
61: #define signo(x) ((x<0)?-1:((x>0)?1:0))
62:
63: //
64: // Defined according to the VBE - VESA 2.0 Standar documentation
65: //
66: struct kVbeInfoBlock {
67:     BYTE    VbeSignature[4];  // DWORD - UDWORD VesaSignatur
68:     BYTE    LVersion;          // Low version
69:     BYTE    HVersion;          // High Version
70:     char    far *CardOEM;        // DWORD to Video Card Sign
71:     DWORD   Capabilities;
72:     WORD    far *ModeList;    // DWORD to Matrix modes
73:     WORD    SizeLFB;          // Granularity (para los Banks)
74:
75:     //---- For VBE 2.0+ --------
76:     WORD   OEMSoftwareRev;    // OEM : Original Equipment Manufacturer
77:     char   far *OEMVendor;    //
78:     char   far *OEMProduct;
79:     char   far *OEMProdVer;
80:     BYTE   Reserved[222];
81:     BYTE   OEMData[256];
82: };
83:
84: struct kVbeModeInfoBlock  { // VideoModeInfo
85:     WORD  ModeAttributes;
86:     BYTE  WinAAttributes;
87:     BYTE  WinBAttributes;
88:     WORD  WinGranularity;
89:     WORD  WinSize;
90:     WORD  WinASegment;
91:     WORD  WinBSegment;
92:
93:     void  far (*WinFuncPtr)();    // ; pointer to window function Bank Switch &
Get Bank
94:     WORD  BytesPerLine;
95:
96:     // Mandatory information for VBE 1.2 and above
```

```
97:     WORD  XResolution;
98:     WORD  YResolution;
99:     BYTE  XCharSize;
100:     BYTE  YCharSize;
101:     BYTE  NumberOfPlanes;
102:     BYTE  BitsPerPixel;
103:     BYTE  NumberOfBanks;
104:     BYTE  MemoryModel;
105:     BYTE  BankSize;
106:
107:     // LBF - Linear Frame Buffer extension
108:     BYTE   LFB_NumPages;
109:     BYTE   LFB_Reserved;
110:
111:     // Direct Color fields (required for direct/6 and YUV/7 memory models)
112:     BYTE  RedMaskSize;
113:     BYTE  RedFieldPosition;
114:     BYTE  GreenMaskSize;
115:     BYTE  GreenFieldPosition;
116:     BYTE  BlueMaskSize;
117:     BYTE  BlueFieldPosition;
118:     BYTE  RsvdMaskSize;
119:     BYTE  RsvdFieldPosition;
120:     BYTE  DirectColorModeInfo;
121:
122:     DWORD  LFB_BaseAdress;
123:     DWORD  OffScreenMemOffset;
124:     DWORD  OffScreenMemSize;
125:     BYTE   Reserved[206];
126: };
127: //------------------------------------------------------------------------------
--
128: struct  kVbeInfoBlock       VbeInfo;
129: struct  kVbeModeInfoBlock VbeMode;
130: char    vbeResult;
131:
132: kVbeInfoBlock         *pVbeInfo = &VbeInfo;  // temporally for send at in ASM
133: kVbeModeInfoBlock      *pVbeMode = &VbeMode;  // same as the previous item
134: //------------------------------------------------------------------------------
135:
136:
137:
138: //------------------------------------------------------------------------------
139: // Dejamos las definiciones de "union" y demas, solo genera errores
140: // ademas de excesivo codigo ASM, un enfoque sencillo, efectivo para
141: // el driver VESA-32BPP a la mas alta resolucion.
142: //------------------------------------------------------------------------------
143: typedef unsigned long COLORREF;
144:
145:
146: COLORREF RGB( UBYTE Red, UBYTE Green, UBYTE Blue, UBYTE Alpha=0 )
147: {
148:     COLORREF color;
149:
150:     // ARGB = 0x00RRGGBB
151:     //
152:     color = Alpha;
153:     color = (color << 8) | Red;
154:     color = (color << 8) | Green;
155:     color = (color << 8) | Blue;
156:
157:     return color;
158: }
```

```
159: //--------------------------------------------------------------------
160: #define kRGB(r,g,b)     RGB(r,g,b)
161: #define kRGBA(r,g,b,a)  RGB(r,g,b,a)
162: //--------------------------------------------------------------------
163: #define GetAValue(rgb)  (UBYTE)(rgb >> 24)
164: #define GetRValue(rgb)  (UBYTE)(rgb >> 16)
165: #define GetGValue(rgb)  (UBYTE)(rgb >>  8)
166: #define GetBValue(rgb)  (UBYTE) rgb
167: //--------------------------------------------------------------------
168: COLORREF kClrGray16[16] ={
169:    0x00000000, 0x00111111, 0x00222222, 0x00333333,
170:    0x00444444, 0x00555555, 0x00666666, 0x00777777,
171:    0x00888888, 0x00999999, 0x00AAAAAA, 0x00BBBBBB,
172:    0x00CCCCCC, 0x00DDDDDD, 0x00EEEEEE, 0x00FFFFFF
173: };
174:
175: COLORREF klrBlack       = 0x00000000;
176: COLORREF klrBlue        = 0x000000BB;
177: COLORREF klrLightBlue   = 0x000000BB;
178: COLORREF klrGreen       = 0x0000BB00;
179: COLORREF klrLightGreen  = 0x0000FF00;
180: COLORREF klrGray        = 0x00808080;
181: COLORREF klrLightGray   = 0x00EDECEB;
182: COLORREF klrLightGray   = 0x00E0E0E0;
183: COLORREF klrDarkGray    = 0x003D3D3D;
184: COLORREF klrWhite       = 0x00FFFFFF;
185:
186:
187: //--------------------------------------------------------------------
188: // VideoAgent : Starts
189: //--------------------------------------------------------------------
190: struct kMode{
191:     int  BPP;          // bits per pixel = 32
192:     int  Width;        // Screen X
193:     int  Height;       // Screen Y
194:     int  BytesPL;      // actual bytes to multip.
195:     int  ModeID;       // To Swith at..
196:     long LFB;          // LinearFB on 386-PMode
197: };
198:
199: kMode LstModes[10] = {
200: // { 24,  640, 480, 0, 0 },  // to hard use 24 on 3bytes
201:   { 32,  640, 480, 0, 0, 0x00 },  // 1
202:   { 32,  800, 600, 0, 0, 0x00 },  // 2
203:   { 32, 1024, 768, 0, 0, 0x00 },  // 3
204: };
205:
206: // wich one we suggest
207: int kvBestMode = 2;
208: int kvBytesPP  = 4;  // Bytes por Pixel
209:
210: kMode kvMode;
211: int   kvBanks;
212:
213:
214: ///------------------------------------------------------------------------
215: /// Prototypes
216: ///------------------------------------------------------------------------
217:
218: // VideoAgent Stuffs
219: //
220: void vAgFindVideoMode();
221: void vAgListVbeModes( WORD nBPP );
```

```
222:
223:
224: // KrystalView Stuffs
225: void kvSetVideoMode( int Mode );
226: void kvSetBank( int Plano );
227:
228:
229: // Basic Draw Functions
230: void kvInitGraph();
231: void kvCloseGraph();
232: void kvSetBackground( COLORREF Color );
233: void kvPutPixel( int iLeft, int iTop, COLORREF Color );
234: void kvLine( int x1, int y1, int x2, int y2, COLORREF Color );
235: void kvRect( int x1, int y1, int x2, int y2, COLORREF Color );
236: void kvRectXor( int x1, int y1, int x2, int y2 );
237: void kvBar( int x1, int y1, int x2, int y2, COLORREF Color );
238: int kvGetMaxWidth();
239: int kvGetMaxHeight();
240: COLORREF kvGetPixel( int iLeft, long iTop );
241:
242:
243: // Extra Draw from bp-Kronoz
244: void kvDrawBitmap( int iLeft, int iTop, char *lpFile );
245: void kvBlendBar( int, int, int, int, COLORREF, float blend = 0.30 );
246:
247:
248: ///-------------------------------------------------------------------------
249: /// Definition's to don't use .cpp file that it doesn't matter
250: ///-------------------------------------------------------------------------
251: void vAgFindVideoMode()
252: {
253:     #ifdef __SMALL__
254:       asm{
255:         mov  ax, 0x4F00
256:         les  di, dword ptr [ pVbeInfo ]
257:         mov  bx, ds
258:         mov  es, bx  // Forzamos
259:         int  0x10
260:         mov  vbeResult, AL
261:       };
262:     #endif
263:
264:
265:     #if defined __LARGE__
266:       asm{
267:         mov  ax, 0x4F00
268:         les  di, [ pVbeInfo ]
269:         int  0x10
270:         mov  vbeResult, AL
271:       };
272:     #endif
273:
274:
275:     textattr( WHITE );
276:     clrscr();
277:
278:     textcolor( LIGHTGREEN );
279:     cprintf("\r\n Av-Fenix VideoAgent 0.1.c Loaded !" );
280:     cprintf("\r\n [ KrystalView | KronozOS Project ]" );
281:     textcolor( WHITE );
282:
283:     if( vbeResult != 0x4F )
284:     {
```

```
285:          printf( "\n\nPC without VESA-Suport" );
286:          return;
287:      }
288:
289:      printf("\n\n Video : %s %d.%d",
290:              pVbeInfo->VbeSignature,
291:              pVbeInfo->HVersion,
292:              pVbeInfo->LVersion );
293:
294:      //
295:      // (char far *)VesaBuffer.CardOEM;   "%Fs"
296:      //
297:      printf("\n OEM String %04X:%04X : %Fs",
298:          FP_SEG(pVbeInfo->CardOEM),
299:          FP_OFF(pVbeInfo->CardOEM),
300:          pVbeInfo->CardOEM );
301:
302:      //printf( "\n" );    vAgListVbeModes( 24 );
303:      printf( "\n" );
304:      vAgListVbeModes( 32 );
305: }
306:
307: void vAgListVbeModes( WORD nBPP )
308: {
309:      unsigned TargetMode;
310:      int      Ind = 0;
311:      int      ModesCont = 0;
312:
313:      // search in all modes collection to find the end
314:      //
315:      for ( ; ( TargetMode=pVbeInfo->ModeList[Ind] ) != 0xFFFF
316:            ; Ind++
317:          )
318:      {
319:          //
320:          // LES DI,VAR
321:          // actualizar  ES:DI  juntos
322:          //
323:          // pero con TASM no se.. no ensambla
324:          //
325:
326:        #ifdef __SMALL__
327:          // _ES = FP_SEG( &Modus );  // struct
328:          _ES = FP_SEG( pVbeMode );  // *
329:          _DI = FP_OFF( pVbeMode );  // *
330:          _CX = TargetMode;
331:          _AX = 0x4F01;
332:          geninterrupt( 0x10 );
333:        #endif
334:
335:        #if defined __LARGE__
336:          asm{
337:              mov  ax, 0x4F01
338:              mov  cx, TargetMode
339:              les  di, [pVbeMode]
340:              int  0x10
341:          };
342:        #endif
343:
344:          // restringir solo a modos de video adecuados
345:          //
346:          if( pVbeMode->BitsPerPixel == nBPP &&
347:              pVbeMode->XResolution >=  640  &&
```

```
348:             pVbeMode->YResolution >=  480 )
349:         {
350:             if( ModesCont < 10 )
351:             {
352:                 LstModes[ModesCont].BPP     = pVbeMode->BitsPerPixel;
353:                 LstModes[ModesCont].LFB     = pVbeMode->LFB_BaseAdress;
354:                 LstModes[ModesCont].ModeID  = TargetMode;
355:                 LstModes[ModesCont].Width   = pVbeMode->XResolution;
356:                 LstModes[ModesCont].Height  = pVbeMode->YResolution;
357:                 LstModes[ModesCont].BytesPL = pVbeMode->BytesPerLine;
358:                 ModesCont++;
359:             }
360:         }
361:     }
362:
363:     // now we should order it all
364:     for( int i=0; i<ModesCont;i++ )
365:       for( int j=i+1; j<ModesCont;j++ )
366:         if( (LstModes[i].Width+LstModes[i].Height) >
367:             (LstModes[j].Width+LstModes[j].Height) )
368:         {
369:             kMode tMode = LstModes[i];
370:             LstModes[i] = LstModes[j];
371:             LstModes[j] = tMode;
372:         }
373:
374:     // localizar y establecer el modo
375:
376:     // stick to the out modes
377:     if( kvBestMode<=0 || kvBestMode>ModesCont )
378:         kvBestMode = 2;
379:
380:     textcolor( LIGHTGREEN );
381:     for( Ind=0; Ind<ModesCont; Ind++ )
382:     {
383:         if( kvBestMode-1 == Ind )
384:             cprintf( "\n\r %d ", Ind+1 );
385:         else
386:             printf( "\n %d ", Ind+1 );
387:
388:         printf( "| 0x%04X  %d-BPP | %4d x %4d  BytesPL:%4d  LFB:%04X:%04X",
389:             LstModes[Ind].ModeID,
390:             LstModes[Ind].BPP,
391:             LstModes[Ind].Width,
392:             LstModes[Ind].Height,
393:             LstModes[Ind].BytesPL,
394:             FP_SEG(LstModes[Ind].LFB),
395:             FP_OFF(LstModes[Ind].LFB)
396:         );
397:
398:         if( kvBestMode-1 == Ind )
399:         {
400:             // select mode by FnxVesaAgent
401:             kvMode  = LstModes[Ind];
402:
403:             // convert 32 to 16 and back
404:             kvBanks = ( ( (long)LstModes[Ind].Width *
405:                           (long)LstModes[Ind].Height *
406:                           (long)kvBytesPP ) / 65536 ) + 1;
407:             //
408:             // Width*Height*BytesPerPixel4 / 64kb
409:             //
410:
```

```
411:             cprintf( " [%d Banks]", kvBanks );
412:         }
413:     }
414: }
415:
416: ///-------------------------------------------------------------------------
417: // Banco o Plano cada uno de 64K 65536 octetos
418: //
419: unsigned int  kLastBank = 1;
420: ///-------------------------------------------------------------------------
421: void kvInitGraph()
422: {
423:     // vAgFindVideoMode();
424:     kvSetVideoMode( KV_VESA32 );
425: }
426: ///-------------------------------------------------------------------------
427: void kvCloseGraph()
428: {
429:     kvSetVideoMode( KV_C8025 );
430: }
431: ///-------------------------------------------------------------------------
432: void kvSetBackground( COLORREF Color )
433: {
434:     //
435:     // We fill the Memory block in one shut
436:     //
437:     for (int i=0; i<kvBanks; i++)
438:     {
439:         kvSetBank( i ),
440:         kSetMemDWord( (void far*) 0xA000, Color, 0x4000 );  // 16384*4 = 65535
441:     }
442:
443:     // Byte  | B | G | R | A |
444:     //       +---------------+
445:     //       | 0 | 1 | 2 | 3 |
446:     //
447:     // ((Color.Blue<<8 | Color.Red)<<8 | Color.Blue)>>8 | Alpha
448:
449:     // DWORD | A | R | G | B |
450: }
451: ///-------------------------------------------------------------------------
452: void kvPutPixel( int iLeft, int iTop, COLORREF Color )
453: {
454:     // para ahorra procesador
455:     //
456:     if( iLeft < 0 || iLeft >= kvMode.Width ) return;
457:     if( iTop < 0 || iTop >= kvMode.Height ) return;
458:
459:     //
460:     // al usar 4Bytes la distribucion de pixels se hace simetrica
461:     // 4 bytes y son multiplo de 65536 lo que facilita el trabajo
462:
463:
464:     long lTop      = iTop; // 32bits -- 16 al divir / 65536
465:
466:     // BitsPP  = 32
467:     // BytesPP = 4
468:
469:     //unsigned int Offset = (iTop*kvMode.BytesPL)+(iLeft*4);
470:     unsigned int Offset = (iTop*kvMode.Width)+(iLeft);
471:     unsigned int NoBank = (lTop*kvMode.BytesPL)+(iLeft*kvBytesPP) >> 16;
472:
473:     if( kLastBank != NoBank )
```

```
474:     {
475:        kvSetBank( NoBank );
476:        kLastBank = NoBank;
477:     }
478:
479:     kVGA[ Offset ] = Color;
480: }
481: ///-------------------------------------------------------------------------
482: COLORREF kvGetPixel( int iLeft, long iTop )
483: {
484:     long      lTop = iTop; // 32bits -- 16 al divir / 65536
485:
486:     // se reduce al no usar CHAR sino LONG
487:     unsigned int Offset = (iTop*kvMode.Width)+(iLeft);
488:     unsigned int NoBank = (lTop*kvMode.BytesPL)+(iLeft*kvBytesPP) >> 16;
489:
490:     if( kLastBank != NoBank )
491:     {
492:        kvSetBank( NoBank );
493:        kLastBank = NoBank;
494:     }
495:
496:     return  kVGA[ Offset ];
497: }
498: ///-------------------------------------------------------------------------
499: void kvLine( int x1, int y1, int x2, int y2, COLORREF Color )
500: {
501:     int dx = x2-x1; // distancia horizontal
502:     int dy = y2-y1; // distancia vertical
503:
504:     int dxabs = abs(dx);
505:     int dyabs = abs(dy);
506:     int sdx   = signo(dx);
507:     int sdy   = signo(dy);
508:
509:     float pendiente;
510:     int i;
511:     if(dxabs>=dyabs) // horizontales y curvas
512:     {
513:         pendiente = (float)dy / (float)dx;
514:         for ( i=0; i!=dx; i+=sdx )
515:             kvPutPixel( i+x1, (pendiente*i)+y1, Color );
516:
517:     }// verticales
518:     else
519:     {
520:         pendiente = (float)dx / (float)dy;
521:         for ( i=0; i!=dy; i+=sdy )
522:             kvPutPixel( (pendiente*i)+x1, i+y1, Color );
523:     }
524: }
525: ///-------------------------------------------------------------------------
526: void kvRect( int x1, int y1, int x2, int y2, COLORREF Color )
527: {
528:     int Ind;
529:
530:     for( Ind=x1; Ind<=x2; Ind++ )
531:     {
532:        kvPutPixel( Ind, y1, Color ); // ---
533:        kvPutPixel( Ind, y2, Color ); // ---
534:     }
535:
536:     for( Ind=y1; Ind<=y2; Ind++ )
```

```
537:     {
538:        kvPutPixel( x1, Ind, Color );// | |
539:        kvPutPixel( x2, Ind, Color );// | |
540:     }
541: }
542: ///--------------------------------------------------------------------------
543: void kvRectXor( int x1, int y1, int x2, int y2 )
544: {
545:     COLORREF Color;
546:     int nValue = 0xBB;
547:
548:
549:     for( int Ind=x1; Ind<=x2; Ind++ )
550:     {
551:        Color = kvGetPixel( Ind, y1 );
552:        kvPutPixel( Ind, y1, kRGB(
553:             GetRValue(Color)^nValue,
554:             GetGValue(Color)^nValue,
555:             GetBValue(Color)^nValue)
556:         ); // ---
557:
558:        Color = kvGetPixel( Ind, y1+1 );
559:        kvPutPixel( Ind, y1+1, kRGB(
560:             GetRValue(Color)^nValue,
561:             GetGValue(Color)^nValue,
562:             GetBValue(Color)^nValue)
563:         ); // ---
564:
565:        Color = kvGetPixel( Ind, y2 );
566:        kvPutPixel( Ind, y2, kRGB(
567:             GetRValue(Color)^nValue,
568:             GetGValue(Color)^nValue,
569:             GetBValue(Color)^nValue)
570:         ); // ---
571:         Color = kvGetPixel( Ind, y2-1 );
572:        kvPutPixel( Ind, y2-1, kRGB(
573:             GetRValue(Color)^nValue,
574:             GetGValue(Color)^nValue,
575:             GetBValue(Color)^nValue)
576:         ); // ---
577:     }
578:
579:     for( Ind=y1+2; Ind<y2-1; Ind++ )
580:     {
581:        Color = kvGetPixel( x1, Ind );
582:        kvPutPixel( x1, Ind, kRGB(
583:             GetRValue(Color)^nValue,
584:             GetGValue(Color)^nValue,
585:             GetBValue(Color)^nValue)
586:         );// | |
587:         Color = kvGetPixel( x1+1, Ind );
588:        kvPutPixel( x1+1, Ind, kRGB(
589:             GetRValue(Color)^nValue,
590:             GetGValue(Color)^nValue,
591:             GetBValue(Color)^nValue)
592:         );// | |
593:
594:        Color = kvGetPixel( x2, Ind );
595:        kvPutPixel( x2, Ind, kRGB(
596:             GetRValue(Color)^nValue,
597:             GetGValue(Color)^nValue,
598:             GetBValue(Color)^nValue)
599:         );// | |
```

```
600:         Color = kvGetPixel( x2-1, Ind );
601:         kvPutPixel( x2-1, Ind, kRGB(
602:             GetRValue(Color)^nValue,
603:             GetGValue(Color)^nValue,
604:             GetBValue(Color)^nValue)
605:         );// | |
606:     }
607: }
608: ///-------------------------------------------------------------------------
609: void kvBar( int x1, int y1, int x2, int y2, COLORREF Color )
610: {
611:     for( int iFil=y1; iFil<=y2; iFil++ )
612:         for( int iCol=x1; iCol<=x2; iCol++ )
613:             kvPutPixel( iCol, iFil, Color );
614: }
615: ///-------------------------------------------------------------------------
616: int kvGetMaxWidth()
617: {
618:     return kvMode.Width;
619: }
620: ///-------------------------------------------------------------------------
621: int kvGetMaxHeight()
622: {
623:     return kvMode.Height;
624: }
625: ///-------------------------------------------------------------------------
626: void  kvDrawBitmap( int iLeft, int iTop, char *lpFile )
627: {
628:     FILE  *fpBmp = fopen( lpFile, "rb" );
629:     if( !fpBmp ) return;
630:
631:     int   nColors;
632:     int      nWidth;
633:     int      nHeight;
634:     int      nRow, nCol;
635:
636:     fseek( fpBmp , 10L, SEEK_SET );
637:     fread( &nColors, sizeof(int), 1, fpBmp );
638:
639:     fseek( fpBmp , 18L, SEEK_SET );
640:     fread( &nWidth, sizeof(int), 1, fpBmp );
641:
642:     fseek( fpBmp , 22L, SEEK_SET );
643:     fread( &nHeight, sizeof(int), 1, fpBmp );
644:
645:
646:     fseek( fpBmp , 54L, SEEK_SET ); // DATA
647:
648:     //printf( "\n Colors = %d ( %d x %d ) ", nColors, nWidth, nHeight );
649:     //  118 = 0x076 = 16Colores  [ no ! ]
650:     //   54 = 0x036 = 24bits
651:     // 1078 = 0x436 = 256Colores
652:
653:     if( nColors == 0x436 ) // 256 Colours
654:     {
655:         unsigned char Pallete[256][4]; // 1024 1Kb
656:         fread( &Pallete, sizeof(Pallete), 1, fpBmp );
657:
658:         for( nRow=0;nRow<nHeight;nRow++ )
659:           for( nCol=0;nCol<nWidth;nCol++ )
660:           {
661:             unsigned char  IndPal = fgetc( fpBmp );
662:             kvPutPixel(iLeft+nCol, ((iTop+nHeight)-nRow)-1,
```

```
663:                 kRGB( Pallete[IndPal][2],Pallete[IndPal][1],Pallete[IndPal][0]
)  );
664:           }
665:     }
666:     else if( nColors == 0x036 ) // 24 Bits
667:     {
668:         for( nRow=0;nRow<nHeight;nRow++ )
669:           for( nCol=0;nCol<nWidth;nCol++ )
670:             kvPutPixel(iLeft+nCol, ((iTop+nHeight)-nRow)-1,
671:                 kRGB( fgetc(fpBmp), fgetc(fpBmp), fgetc(fpBmp) ) );
672:
673:     }
674:
675:     fclose( fpBmp );
676: }
677: ///-------------------------------------------------------------------------
------------
678: void  kvDrawBmpIcon( int iLeft, int iTop, char *lpFile )
679: {
680:     FILE  *fpBmp = fopen( lpFile, "rb" );
681:     if( !fpBmp ) return;
682:
683:     int   nColors;
684:     int      nWidth;
685:     int      nHeight;
686:     int      nRow, nCol;
687:
688:     fseek( fpBmp , 10L, SEEK_SET );
689:     fread( &nColors, sizeof(int), 1, fpBmp );
690:
691:     fseek( fpBmp , 18L, SEEK_SET );
692:     fread( &nWidth, sizeof(int), 1, fpBmp );
693:
694:     fseek( fpBmp , 22L, SEEK_SET );
695:     fread( &nHeight, sizeof(int), 1, fpBmp );
696:
697:
698:     fseek( fpBmp , 54L, SEEK_SET ); // DATA
699:
700:     //printf( "\n Colors = %d ( %d x %d ) ", nColors, nWidth, nHeight );
701:     //  118 = 0x076 = 16Colores  [ no ! ]
702:     //   54 = 0x036 = 24bits
703:     // 1078 = 0x436 = 256Colores
704:
705:     if( nColors == 0x436 ) // 256 Colours
706:     {
707:         unsigned char Pallete[256][4]; // 1024 1Kb
708:         fread( &Pallete, sizeof(Pallete), 1, fpBmp );
709:
710:         for( nRow=0;nRow<nHeight;nRow++ )
711:           for( nCol=0;nCol<nWidth;nCol++ )
712:           {
713:             unsigned char  IndPal = fgetc( fpBmp );
714:             if( IndPal != 0 ) // aplicar mascara 0 - Magenta
715:                 kvPutPixel(iLeft+nCol, ((iTop+nHeight)-nRow)-1,
716:                     kRGB(
Pallete[IndPal][2],Pallete[IndPal][1],Pallete[IndPal][0] )  );
717:           }
718:     }
719:     /*
720:     else if( nColors == 0x036 ) // 24 Bits
721:     {
722:         for( nRow=0;nRow<nHeight;nRow++ )
```

```
723:            for( nCol=0;nCol<nWidth;nCol++ )
724:              kvPutPixel(iLeft+nCol, ((iTop+nHeight)-nRow)-1,
725:                  kRGB( fgetc(fpBmp), fgetc(fpBmp), fgetc(fpBmp) ) );
726:
727:      }
728:      */
729:
730:      fclose( fpBmp );
731: }
732: ///-------------------------------------------------------------------------
------------
733: void kvBlendBar( int x1, int y1, int x2, int y2, COLORREF Color, float blend
)
734: {
735:      for( int iFil=y1; iFil<=y2; iFil++ )
736:      {
737:          for( int iCol=x1; iCol<=x2; iCol++ )
738:          {
739:              COLORREF Back = kvGetPixel(iCol,iFil);
740:
741:              // generado por Interpolacion de Color
742:              //
743:              unsigned char abR = (GetRValue(Back) * blend + (1-blend) *
                                      GetRValue(Color));
744:              unsigned char abG = (GetGValue(Back) * blend + (1-blend) *
                                      GetGValue(Color));
745:              unsigned char abB = (GetBValue(Back) * blend + (1-blend) *
                                      GetBValue(Color));
746:
747:              kvPutPixel( iCol, iFil, kRGB(abR,abG,abB) );
748:          }
749:      }
750: }
751: ///-------------------------------------------------------------------------
752: /*
753: void kvBar16b( WORD Left, WORD Top, WORD Right, WORD Bottom, WORD Color )
754: {
755:
756:      //
757:      // Calculate the Offset (Desplazamineto) in the
758:      // Pointer and set the New Value
759:      //
760:      int SF_CXSCREEN = 800;
761:      int SF_CYSCREEN = 600;
762:      int VESA_SEG    = 0xA000;
763:
764:      WORD  Corner = (SF_CXSCREEN*(DWORD)Top) + (Left);
765:      WORD  Plane  = ( ((DWORD)Top*(SF_CXSCREEN) )+(Left) )>>15;
766:
767:      kvSetBank( Plane );
768:      WORD  nLong = 1+( (Right>=SF_CXSCREEN?SF_CXSCREEN-1:Right) - Left);
769:      WORD  nAlt  = 1+( (Bottom>=SF_CYSCREEN?SF_CYSCREEN-1:Bottom) - Top);
770:
771:      WORD  nLongx2=nLong*2;
772:      Corner = Corner*2;
773:
774:
775:      //-------------------------------------------------------------
776:      // kBar : Contemplaciones de area :
777:      // -------------------------------
778:      //
779:      // para cuando los pixels necesiten cambiar de banco
780:      // ------------------------------------------------
```

```
781:     //
782:     //              +--------
783:     //              |      Fila = 102
784:     //  ------------+
785:     //  320x200  127|128   Fila = 103
786:     //
787:     // kBar( 126,100, 129, 100, RGB(0x00,0xFF,200) );
788:     //
789:     //
790:     // Cuando son bancos alternos pero no como el caso ant.
791:     // ---------------------------------------------------
792:     //
793:     //       +---------------------+
794:     //       |                     |    +--------
795:     //       |       Bar-Bank1     |    |      Fila = 102
796:     //  -----------------------------+
797:     //       |       Bar-Bank-2    |
798:     //       +---------------------+
799:     //
800:     //  320x200  127|128   Fila = 103
801:     //
802:     // NOTA : es necesario que sea ASM  pues es la  unica forma
803:     // de acelarar las instrucciones  par  el  Micro-Procesador
804:     // ademas solo en ASM podemos usar (REP STOSW) importante.
805:     //-------------------------------------------------------------
806:
807:     asm{
808:         mov  dx,VESA_SEG ;// VESA Memory
809:         mov  es,dx         ;//
810:         xor  dx,dx       ;// set like 0
811:         mov  ax,Color     ;// the color
812:     };                     ;//
813:     Repaint:             ;// Here for paint
814:     asm{                 ;//
815:         mov     di,Corner     ;// first corner
816:         mov  cx,nLong    ;// How many
817:         mov  bx,di         ;// .
818:         add  bx,nLongx2  ;// .
819:         jnc  PaintBank2         ;// not Overweight over 0xFFFF !
820:                            ;//
821:              cmp  bx,0x0000  ;// is intersection banks
822:              jz   PaintBank2 ;// then skip to here
823:                              ;//
824:              mov  cx,0x0000  ;// 65536+1
825:              sub  cx,di      ;//
826:              shr  cx,1       ;// Ancho/2
827:              mov  bx,cx      ;// save nLong
828:              rep  stosw      ;// Store WORD
829:                              ;// first bank
830:              mov  cx,nLong   ;//
831:              sub  cx,bx      ;//
832:              ;//---------------------------
833:              mov  bx,0x01
834:              jmp  NextBank
835:     };
836:     PaintBank1:
837:     asm{
838:              rep  stosw         ;// Store WORD second bank
839:              mov  bx,SF_CXSCREEN
840:              add  bx,bx      ;// Screen*2
841:              add  Corner,bx     ;//
842:              jmp  NextLine   ;// Jump direct
843:     };
```

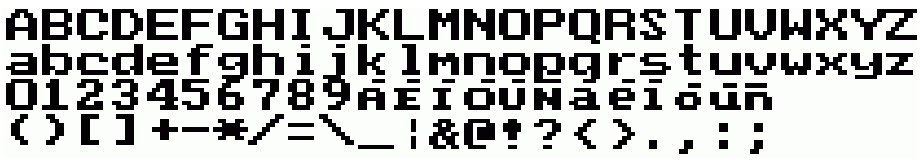
```
844:     PaintBank2:
845:     asm{
846:         rep  stosw          ;// Store WORD first bank
847:         mov  bx,SF_CXSCREEN
848:         add  bx,bx         ;// Screen*2
849:         add  Corner,bx      ;//
850:         jnc  NextLine     ;// Jump if not overflow
851:
852:              mov  bx,0x02
853:              jmp  NextBank
854:     };
855:     NextLine:
856:     asm{
857:         inc  dx              ;// Next line
858:         cmp  dx,nAlt      ;// the finish ?
859:         jnz  Repaint      ;// paint next line
860:         jmp  EndBar       ;// when loop finish
861:     };//-------------------------------------
862:     NextBank:
863:     asm{//----------------------------
864:         pusha             ;// save regs
865:         inc  Plane          ;// next
866:         mov  ax,0x4F05      ;// set Bank
867:         mov  bx,0x0000      ;// Window
868:         mov  dx,Plane      ;// Plane
869:         int  0x10          ;// video int
870:         popa              ;// restore
871:         ;//----------------------------
872:         cmp  bx,0x01
873:         jnz  NextLine      ;// de ser 2
874:         jmp  PaintBank1      ;// de ser 1
875:     };
876:     EndBar:
877:     kvSetBank( Plane );       ;// Update the Bank
878: }
879: */
880:
881: ///-------------------------------------------------------------------------
882: void kvSetVideoMode( int Mode )
883: {
884:     //-------------------------------------------
885:     if( Mode == 0 )  // C8025
886:         asm{
887:           MOV  AX,0x0003  ;// AH=0  AL=03
888:           INT  0x10
889:         };
890:     //-------------------------------------------
891:     if( Mode == 1 ) // VGA16
892:         asm{
893:           MOV  AX, 0x0013  ;// AH=0  AL=13  VGA16
894:           INT  0x10
895:         };
896:     //-------------------------------------------
897:     int vbe32Mode = kvMode.ModeID;
898:     if( Mode == 2 )  // VBE - VESA
899:         asm{
900:           MOV  AX, 0x4F02       ;// VBE 2.0
901:           MOV  BX, vbe32Mode    ;// Setup FnxAg
902:           INT  0x10
903:         };
904:
905: };
906: ///-------------------------------------------------------------------------
```

```
907: void kvSetBank( int Plano )
908: {
909:     asm{
910:         MOV  AX, 0x4F05
911:         MOV  BX, 0x00
912:         MOV  DX, Plano
913:         INT  0x10
914:     };
915: };
916: ///--------------------------------------------------------------------------
917: #endif
918:
```

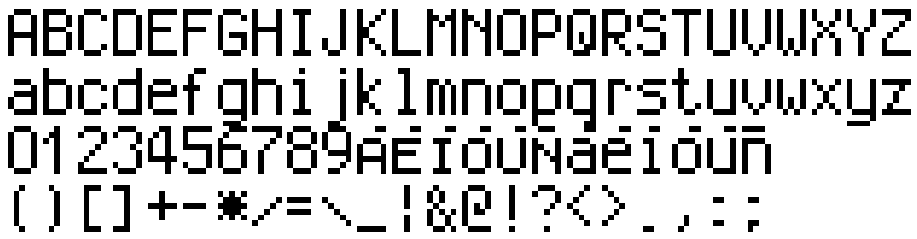
```
01:
02: /*************************************************************************
03:  * Kronoz Project - Av-Fenix Antivirus
04:  * Modulo : KrystalView - Memory Functions on 32bits
05:  * Fecha  : Febrero de 2012
06:  *
07:  *************************************************************************/
08:
09: #if !defined _KrystalMemoryManager_H
10: #define _KrystalMemoryManager_H
11:
12: // En el compilador BCC52 far no importa, solo el modelo de memoria
13: //
14: void kSetMemWord( void far *pSegment, WORD iVal, UWORD iCont );
15: void kSetMemDWord( void far *pSegment, DWORD lVal, UDWORD lCont );
16:
17: void kSetMemWord( void far *pSegment, WORD iVal, UWORD iCont )
18: {
19:     asm{
20:         MOV AX, pSegment  ;// 0xB800:0000
21:         MOV ES, AX
22:         MOV DI, 0         ;// Inicio
23:         MOV CX, iCont     ;// Cont
24:         MOV AX, iVal      ;// Value
25:
26:         REP STOSW         ;// set 16bits
27:     };
28: }
29:
30: void kSetMemDWord( void far *pSegment, DWORD lVal, UDWORD lCont )
31: {
32:     asm{
33:         MOV EAX, pSegment
34:         MOV  ES, EAX
35:         MOV  DI, 0
36:         MOV ECX, lCont
37:         MOV EAX, lVal
38:
39:         REP STOSD
40:     };
41: }
42:
43: #endif
44:
```

Generacion de Fuentes en KrystalView

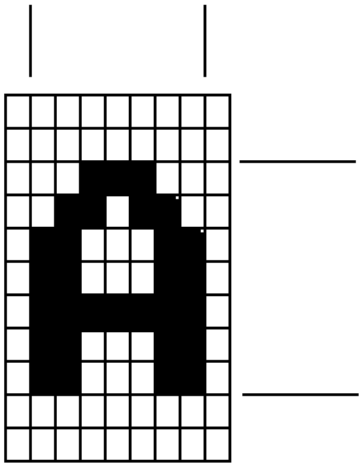
Se tiene un archivo BMP que se mapea en tiempo de ejecución y logra desplegar letras basadas en su poción matricial, tenemos dos tipos de letra usados en el enorno KrystalView.



Fuente BOLD de 8x8 por fuente.



Fuente Menuet de 7x8 por fuente.



```
01:
02: /**************************************************************************
03:  * Kronoz Project - Av-Fenix Antivirus
04:  * Modulo : KrystalView - Font Set & DrawText
05:  * Fecha  : Febrero de 2012
06:  *
07:  * Historial :
08:  *
09:  *   Febrero 17 : Fuentes BMP imagen de 192x62 7x13 --> 7x14 Reales
10:  *                Simplificaion y optimizacion de COLORREF - RGB
11:  *
12:  **************************************************************************/
13:
14:
15: #if !defined _KrystalViewFonts_H
16: #define _KrystalViewFonts_H
17:
18: #include <string.h>
19:
20: //---------------------------------------------------------------------------
21: void kvSetFontColor( COLORREF Color );
22: void kvDrawText( int iLeft, int iTop, char *lpString );
23:
24:
25: /*
26: unsigned char keyMap[] = { = ASCII
27:   "                    "  //  0 - 19
28:   "                    "  // 20 - 39
29:   "                    "  // 40 - 59
30:   "<=>?@ABCDEFGHIJKLMNO"  // 60 - 79
31:   "PQRSTUVWXYZ[\\]^_'abc" // 80 - 99
32:   "defghijklmnopqrstuvw" // 100 - 119
33:   "xyz{|}" // 120 - 139
34: };
35: */
36:
37: //---------------------------------------------------------------------------
38: unsigned char KeyMap[] = {
39:   "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
40:   "abcdefghijklmnopqrstuvwxyz"
41:   "0123456789ÁÉÍÓÚÑáéíóúñ"
42:   "()[]+-*/=\\_|&@!?<>.,:;"
43: };
44:
45: /*
46: #define FNT_BMP_NAME "fnVerdn.bmp"
47:
48: #define FNT_BMP_WIDTH  184
49: #define FNT_BMP_HEIGHT  54
50:
51: #define fntWidth   7
52: #define fntHeight 13
53:
54:
55: struct kFont{
56:     unsigned char Car;
57:     unsigned char BMP[fntHeight][fntWidth];
58: };
59: //---------------------------------------------------------------------------
60: kFont CharMap[101] = {
61:   32, // " " Espacio en Blanco
62:   1, 1, 1, 1, 1, 1, 1,
63:   1, 1, 1, 1, 1, 1, 1,
```

```
64:   1, 1, 1, 1, 1, 1, 1,
65:   1, 1, 1, 1, 1, 1, 1,
66:   1, 1, 1, 1, 1, 1, 1,
67:   1, 1, 1, 1, 1, 1, 1,
68:   1, 1, 1, 1, 1, 1, 1,
69:   1, 1, 1, 1, 1, 1, 1,
70:   1, 1, 1, 1, 1, 1, 1,
71:   1, 1, 1, 1, 1, 1, 1,
72:   1, 1, 1, 1, 1, 1, 1,
73:   1, 1, 1, 1, 1, 1, 1,
74:   1, 1, 1, 1, 1, 1, 1,
75: };
76: */
77:
78: /*
79: #define FNT_BMP_NAME "fnBorlnd.bmp"
80:
81: #define FNT_BMP_WIDTH  208
82: #define FNT_BMP_HEIGHT  44
83:
84: #define fntWidth  8
85: #define fntHeight 8
86: */
87:
88: #define FNT_BMP_NAME "fnMenuet.bmp"
89:
90: #define FNT_BMP_WIDTH  160
91: #define FNT_BMP_HEIGHT  40
92:
93: #define fntWidth   6
94: #define fntHeight 10
95:
96:
97: struct kFont{
98:     unsigned char Car;
99:     unsigned char BMP[fntHeight][fntWidth];
100: };
101: //------------------------------------------------------------------------------
---
102: kFont CharMap[101] = {
103:   32, // " " Espacio en Blanco
104:   1, 1, 1, 1, 1, 1, 1, 1,
105:   1, 1, 1, 1, 1, 1, 1, 1,
106:   1, 1, 1, 1, 1, 1, 1, 1,
107:   1, 1, 1, 1, 1, 1, 1, 1,
108:   1, 1, 1, 1, 1, 1, 1, 1,
109:   1, 1, 1, 1, 1, 1, 1, 1,
110:   1, 1, 1, 1, 1, 1, 1, 1,
111:   1, 1, 1, 1, 1, 1, 1, 1,
112: };
113: //------------------------------------------------------------------------------
114: kFont         kTmpCar;
115: COLORREF     kFontColor = klrBlack;
116: //------------------------------------------------------------------------------
117:
118: void kvFontLoad()
119: {
120:     for( int i=1; i<=100; i++ )
121:         CharMap[i].Car = 0;
122:
123:     FILE  *fpBmp = fopen( FNT_BMP_NAME, "rb" );
124:     if( !fpBmp ) return;
125:
```

```
126:     char *pMemBmp = new char [FNT_BMP_WIDTH*FNT_BMP_HEIGHT];
127:
128:     int      nWidth;
129:     int      nHeight;
130:     int      nRow, nCol;
131:
132:     fseek( fpBmp , 18L, SEEK_SET );
133:     fread( &nWidth, sizeof(int), 1, fpBmp );
134:
135:     fseek( fpBmp , 22L, SEEK_SET );
136:     fread( &nHeight, sizeof(int), 1, fpBmp );
137:
138:     //unsigned char Pallete[256][4]; // 1024 1Kb
139:     //fread( &Pallete, sizeof(Pallete), 1, fpBmp );
140:     fseek( fpBmp , 54+1024, SEEK_SET ); // DATA
141:
142:     // Convertir a BMP en memoria
143:     for( nRow=0;nRow<nHeight;nRow++ )
144:       for( nCol=0;nCol<nWidth;nCol++ )
145:         pMemBmp[ nRow*FNT_BMP_WIDTH + nCol ] = fgetc( fpBmp );
146:
147:     fclose( fpBmp );
148:
149:     //-------------------------------------------------------------------------
150:     // Mapear la memoria y sacar los BMPs
151:     //-------------------------------------------------------------------------
152:     int nMapRow = 0;
153:     int nMapCol = 0;
154:     int nIndCar = 0;
155:
156:     char rowCount[] = { 26, 26, 22, 22 };
157:     for( nMapRow=0; nMapRow<4; nMapRow++ )
158:     {
159:         for( nMapCol=0; nMapCol<rowCount[nMapRow]; nMapCol++ )
160:         {
161:             CharMap[1+nIndCar].Car = KeyMap[ nIndCar ];
162:
163:             // OJO 14 : (fntHeight=13) + 1
164:             for( nRow=(nMapRow*fntHeight); nRow<(nMapRow+1)*fntHeight; nRow++
165:                 for( nCol=(nMapCol*fntWidth); nCol<(nMapCol+1)*fntWidth;
166:                     CharMap[1+nIndCar].BMP[nRow%fntHeight][nCol%fntWidth] =
167:                         pMemBmp[ (nHeight-nRow-1)*FNT_BMP_WIDTH + nCol ];
168:
169:             nIndCar++;
170:         }
171:     }
172:     //-------------------------------------------------------------------------
173:     delete pMemBmp;
174: }
175: //-----------------------------------------------------------------------------
176: void fntSetBmpChar( char Charset )
177: {
178:     // espacio en blanco
179:     kTmpCar = CharMap[0];
180:
181:     for( int i=0; i<=100; i++ )
182:     {
183:         // BMP de caracter seleccionado
184:         if( CharMap[i].Car == Charset )
185:             kTmpCar = CharMap[i];
186:     }
187: }
188: //-----------------------------------------------------------------------------
```

```
189: void fntDrawChar( int x, int y, char C )
190: {
191:     int i, j;
192:
193:     fntSetBmpChar( C );
194:     for( i=0; i<fntHeight; i++ )
195:         for( j=0; j<fntWidth; j++ )
196:         {
197:             if( kTmpCar.BMP[i][j] == 0 )
198:                 kvPutPixel( x+j, y+i, kFontColor );
199:         }
200: }
201: //-----------------------------------------------------------------------------
202: void kvSetFontColor( COLORREF Color )
203: {
204:     kFontColor = Color;
205: }
206: //-----------------------------------------------------------------------------
207: void kvDrawText( int iLeft, int iTop, char *lpString )
208: {
209:     for( int i=0; i<strlen(lpString); i++ )
210:         fntDrawChar( iLeft + i*fntWidth, iTop, lpString[i] );
211: }
212:
213: #endif
214:
```

Codigo fuente de la Interfaz AvFenix LiveDVD

```
01:
02: /*************************************************************************
03:  * Kronoz Project - Av-Fenix Antivirus
04:  * Modulo : KrystalView Environment + [KronozVideo Agent]
05:  * Fecha  : Febrero de 2012
06:  *
07:  * Compilar : bcc -ml /B /3 kvFenix.cpp
08:  *
09:  *************************************************************************/
10:
11: #include "kvGraph.h"
12: #include "kvFonts.h"
13: #include "kvWidgts.h"
14:
15: #include "FxAbout.h"
16:
17: #define FENIX_DEBUG 1
18:
19:
20: class KvFenixApp : public KApp
21: {
22:     protected:
23:         //KDialog *pDialog;
24:
25:     private:
26:         BOOL OnInit();
27:         void OnExit();
28:
29:     public:
30:         int Result;
31:
32:         void Run();
33: };
34:
35: void FxShowLoad( int Left, int Top, int Delay )
36: {
37:     int i, j;
38:
39:     COLORREF tmp, tail[20] = {
40:            0x00F0F0F0, klrLightGray, klrGray, klrDarkGray, klrWhite,
41:            klrWhite, klrWhite, klrWhite, klrWhite, klrWhite,
42:            klrWhite, klrWhite, klrWhite, klrWhite, klrWhite,
43:            klrWhite, klrWhite, klrWhite, klrWhite, klrWhite,
44:         };
45:
46:     kvRect( Left, Top, Left+202, Top + 16, klrDarkGray );
47:
48:     for( i=0; i<Delay; i++ )
49:     {
50:         for( j=0; j<20; j++ )
51:             kvBar( Left+j*10+2, Top+2, Left+(j+1)*10, Top+14, tail[j] );
52:
53:         delay(50);
54:
55:         tmp = tail[19];
56:         for( j=19; j>=1; j-- )
57:             tail[j] = tail[j-1];
58:         tail[0] = tmp;
59:     }
```

```
60: }
61:
62: BOOL KvFenixApp::OnInit()
63: {
64:     // iniciar los graficos, fuentes y Widgets
65:     vAgFindVideoMode();
66:
67:     #if defined FENIX_DEBUG
68:     delay(2000);
69:     #endif
70:
71:     kvInitGraph();
72:     kvFontLoad();
73:
74:     kvSetBackground( kRGB(252,254,252) );
75:     kvDrawBitmap( (kvGetMaxWidth()-320)/2,
                      (kvGetMaxHeight()-110)/2, "Krystal.bmp" );
76:
77:     #if defined FENIX_DEBUG
78:     //delay(2000);
79:     FxShowLoad( (kvGetMaxWidth()-160)/2, (kvGetMaxHeight()+220)/2, 100 );
80:     #endif
81:
82:
83:     // let's draw a desktop
84:     //kvSetBackground( kRGB(60,120,0) );
85:     kvSetBackground( kRGB(6,57,9) );
86:     kvDrawBitmap( (kvGetMaxWidth()-1024)/2,
                     (kvGetMaxHeight()-768)/2, "Screen.bmp" );
87:
88:     // Charge Bar and dialog
89:     kvSurface( 0, 0, kvGetMaxWidth(), 40 );
90:     kvDrawBmpIcon( 10, 5, "icon0.bmp" );
91:
92:     return TRUE;
93: }
94: //-------------------------------------------------------------------------
95: void KvFenixApp::OnExit()
96: {
97:     kvCloseGraph();
98:     textattr( 10 );    cprintf( "\n\r Fenix Antivirus 2012 on KrystakView");
99:     textattr( 14 );    cprintf( "\n\r Ejecucion finalizada.\n");
100: }
101: //-------------------------------------------------------------------------
102: void KvFenixApp::Run()
103: {
104:     KvFenixApp::OnInit();
105:
106:     KDialog *pCalendar = new KDialog( "CALENDARIO", KW_GADGET );
107:     pCalendar->SetRect( kvGetMaxWidth()-146, 50, 136, 154 );
108:     pCalendar->ShowModal();
109:     kvDrawBitmap( pCalendar->Left+8, pCalendar->Top+26, "Calendar.bmp" );
110:     delete pCalendar;
111:
112:
113:     int   dskChoice;
114:     char *dskMenu[] = {
115:         "Ejecutar Fenix-Antivirus",
116:         "Reiniciar el computador",
117:         "Acerca de los autores",
118:         "Salir del Programa",
119:     };
120:
```

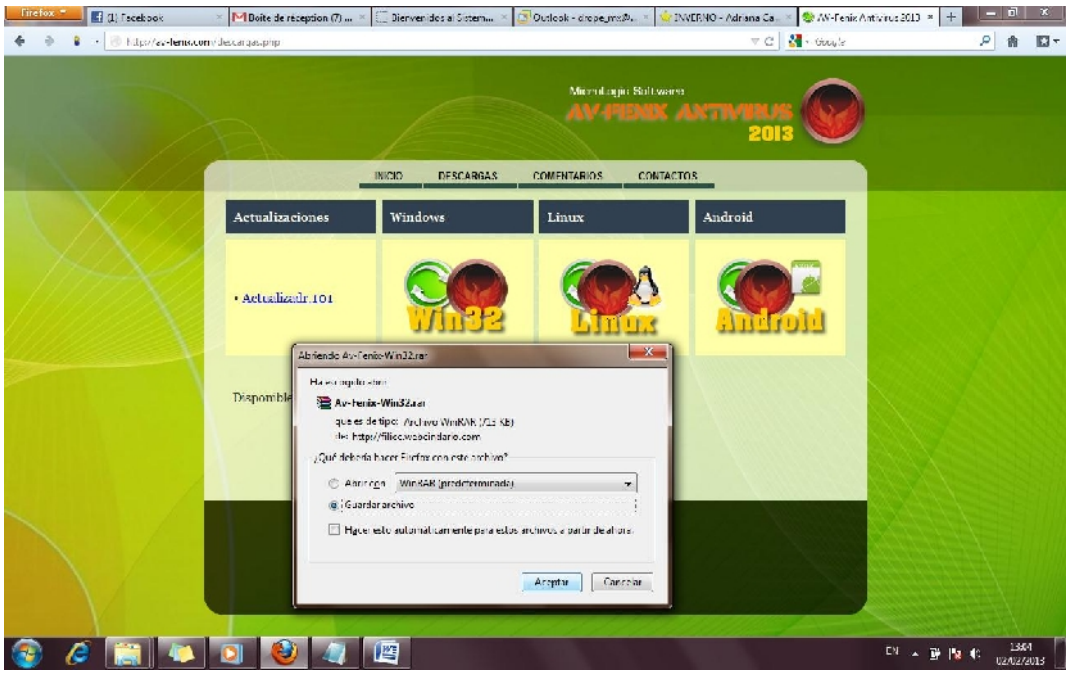
```
121:     do{
122:
123:         dskChoice = kvMenu( 2, 42, dskMenu, 4 );
124:
125:         if( dskChoice == 0 )
126:         {
127:             KDialog *pDialog = new KDialog( "AV-Fenix Antivirus 2012" );
128:             pDialog->SetRect( 10, 50, 480, 420 );
129:             pDialog->SetBackName( "_dlA" );
130:             pDialog->ShowModal();
131:             getch();
132:             delete pDialog;
133:         }
134:         if( dskChoice == 1 )
135:         {
136:             system("reboot.com");
137:         }
138:         if( dskChoice == 2 )
139:         {
140:             KDialog *pAbout = new KDialog( "Sobre Av-Fenix 2012" );
141:             pAbout->SetBkColor( klrBlack );
142:             pAbout->SetBackName( "_dlB" );
143:
144:             pAbout->ShowModal();
145:
146:             kvSetFontColor( klrLightGreen );
147:             kvDrawText( pAbout->Left+20, pAbout->Top+340,
147:                                 "AV-FENIX ANTIVIRUS 2013" );
148:
149:             kvSetFontColor( klrWhite );
150:             kvDrawText( pAbout->Left+20, pAbout->Top+358,
150:                                 "(c) MicroLogic Software Corp - 2012" );
151:             kvDrawText( pAbout->Left+20, pAbout->Top+372,
151:                                 "E.P. Ingeniería Estadística e Informática" );
152:             k3dCube();
153:             delete pAbout;
154:         }
155:
156:     }while( dskChoice != 3 );
157:
158:     KvFenixApp::OnExit();
159: }
160: //------------------------------------------------------------------------
161: int main( int ParamCont, char *Params[] )
162: {
163:     if( ParamCont == 2 )
164:         kvBestMode = Params[1][1] - '0';
165:
166:     KvFenixApp *pFenix  = new KvFenixApp;
167:     pFenix->Run();
168:     delete pFenix;
169:
170:     return 0;
171: }
172:
```

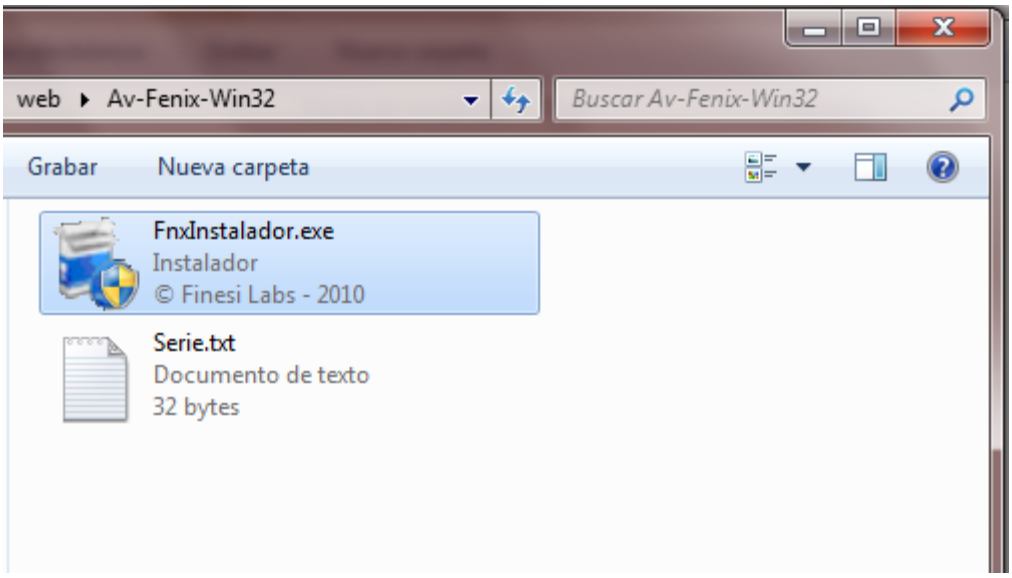
ANEXO E

MANUAL DE USUARIO

1. Descargar Av-Fenix Antivirus 2013

Puede descargarla directamente en la sección de descargas en nuestro portal Web desde http://www.av-fenix.com

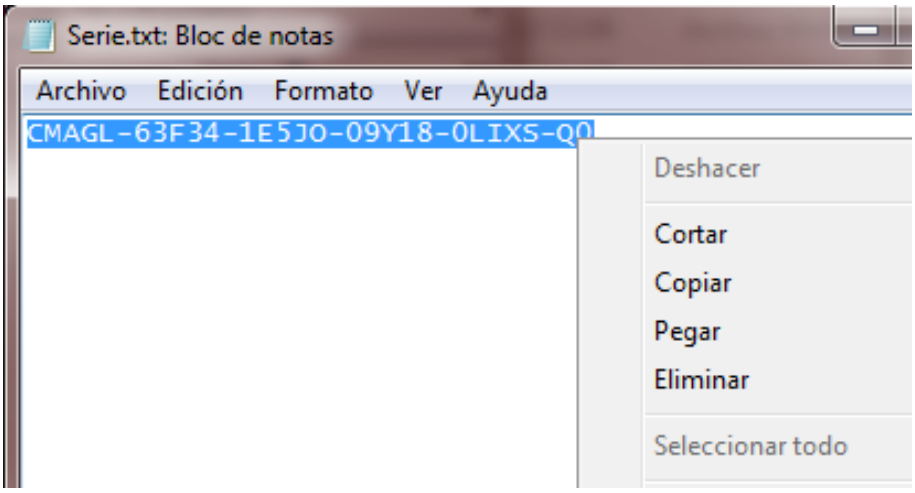




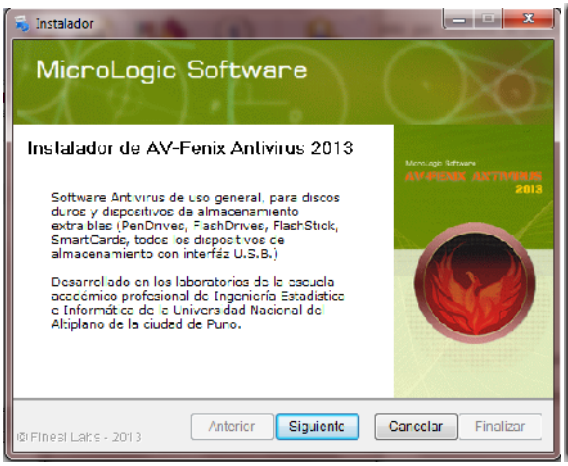
Una vez descargado y descomprimido tendremos dos archivos el primero el Instalador y seguidamente la serie para desencriptar el contenido del archivo extraíble, se ha usado una técnica desarrollada personalmente para crear instaladores sin tener que recurrir a InstallShield.

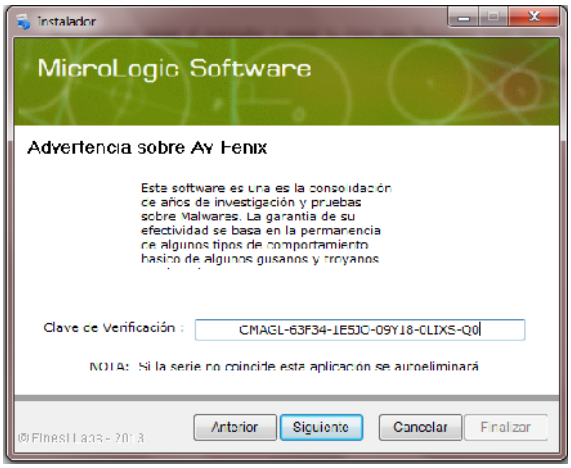
2. Instalar

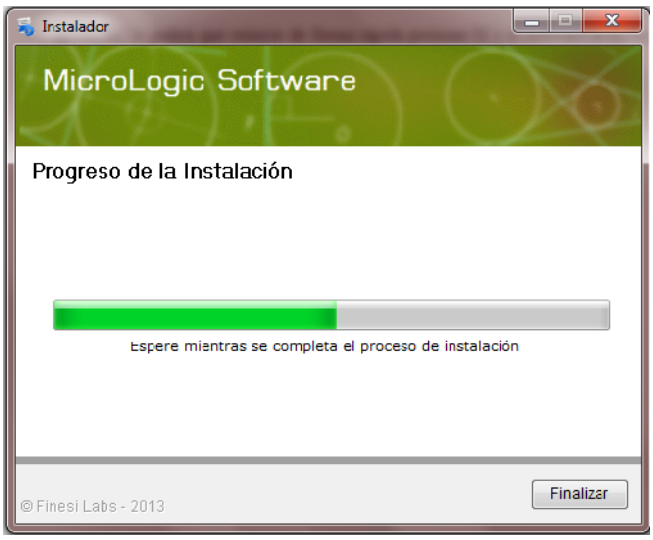
Como primer paso debe abrirse la serie, pues se autodestruirá el contenido si se ejecuta primer el instalador, es un esquema de seguridad que se desarrollo para evitar copias innecesarias y una forma sutil de proteger el producto.

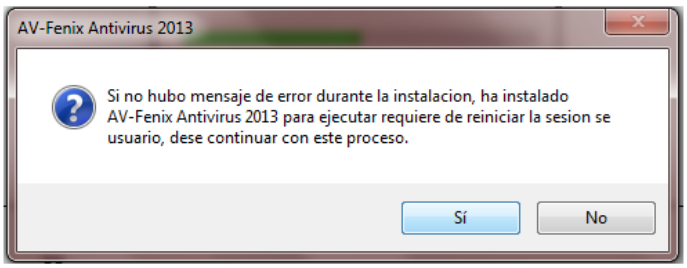


Seguidamente cargar el instalador y pegar la clave para finalmente tener instalado el producto, le pedirá que reinicie de forma rápida presione SI y el software se encargara de cerrar la sesión de Windows





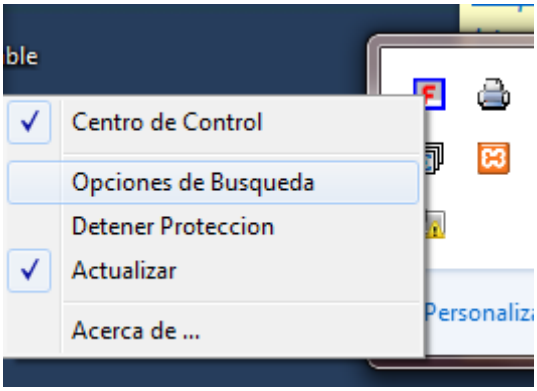




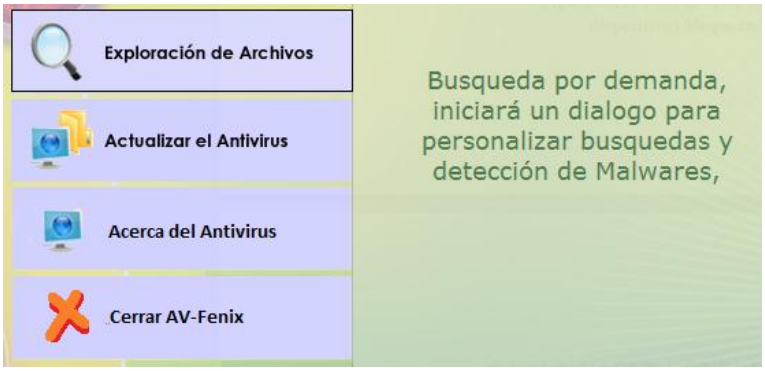
3. Escaneo por Demanda

Esta opción permitirá que realice sus búsquedas de forma manual. **VACUNA ONDEMMAND**

La vacuna **OnDemmand** se le denomina a la vacuna por petición explicita del usuario, cuando requiere una evaluación más exhaustiva.



Modo 1: Por la barra de tareas, click derecho y menú contextual.



Modo 2: Por el Panel de Control, menú AV-Fenix.

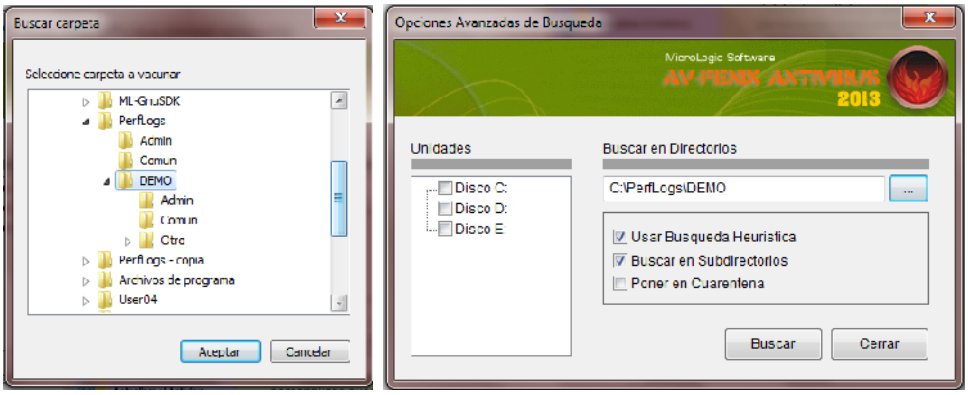
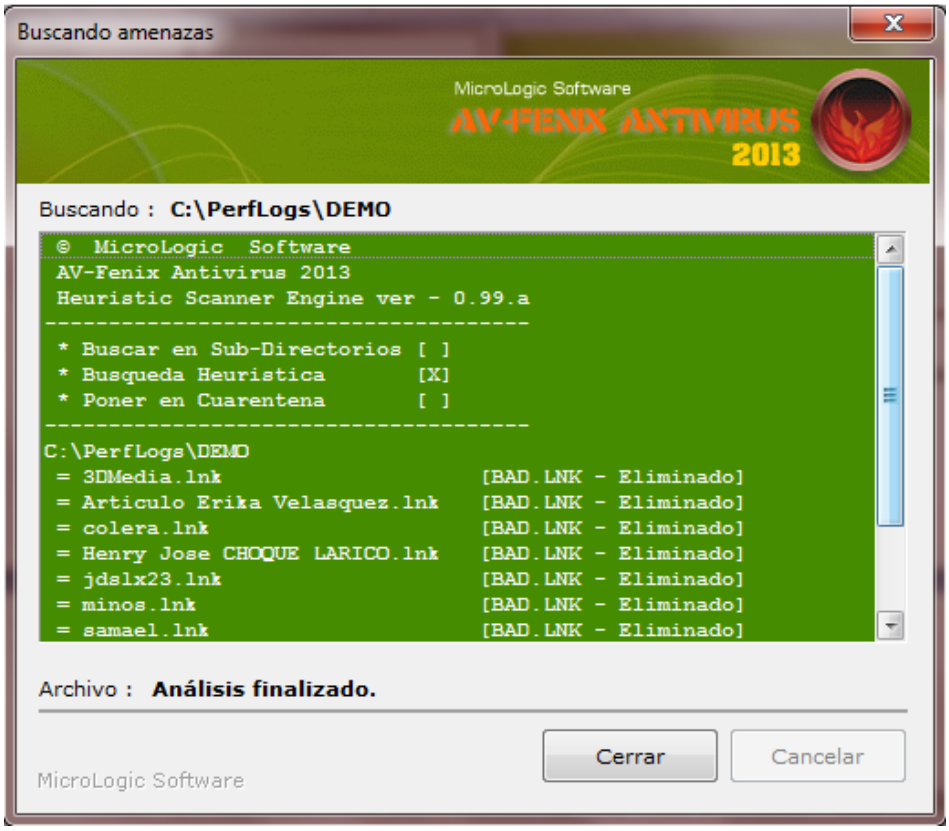


Figura 4.13: Dialogo de selección de Unidad e incluso sub-directorios.



Resultados de la vacunación OnDemmand.

Para la muestra se infecto bajo control la carpeta **C:\PerfLogs\DEMO** con 14 variedades de gusanos y dañado 4 subdirectorios, después de la vacuna **OnDemmand** tenemos los siguientes resultados.

Infección Controlada	Infectados	A Prueba	Limpiados
Gusanos	14	14	13
Carpetas Dañadas	4	4	4
%	--	**100%**	**90%**