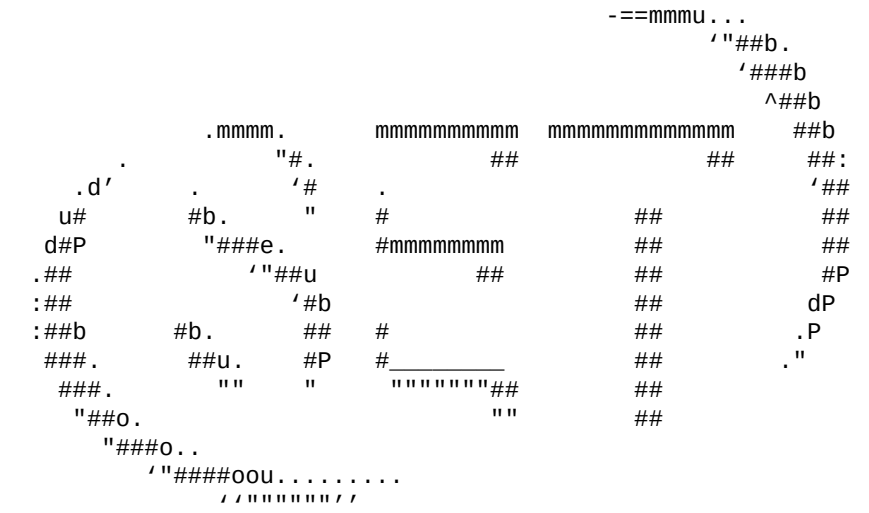


Saqueadores Edicion Tecnica

INFORMACION LIBRE PARA GENTE LIBRE

SET #29 - 16 de Febrero de 2004

```
ú---------------------------------ú--------------------------------------ú
|                                 |                                      |
ú---[ EDITORIAL ]----------------ú                                      |
|                                                                        |
| SET Ezine                                                              |
|                                                                        |
| Disponible en:                                                         |
|     http://www.set-ezine.org                                           |
|                                                                        |
| Mirrors:                                                               |
|     http://salteadores.tsx.org                                         |
|     http://www.hackemate.com.ar/ezines/set/ (¡version on-line!)        |
|                                                                        |
| Contacto:                                                              |
|     <web@set-ezine.org>                                                |
|     <set-fw@bigfoot.com>                                               |
|                                                                        |
|                                                                        |
| Copyright (c) 1996 - 2004 SET - Saqueadores Edicion Tecnica -          |
ú------------------------------------------------------------------------ú


ú----------[ AVISO ]-------------ú--------------------------------------ú
|                                 |                                      |
ú---[ ADVERTENCIAS ]-------------ú                                      |
|                                                                        |
|  * La INFORMACION contenida en este ezine no refleja la opinion de     |
|  nadie y se facilita con caracter de mero entretenimiento, todos       |
|  los datos aqui presentes pueden ser erroneos, malintencionados,       |
|  inexplicables o carentes de sentido.                                  |
|                                                                        |
|  El E-ZINE SET no se responsabiliza ni de la opinion ni de los         |
|  contenidos de los articulos firmados y/o anonimos.                    |
|                                                                        |
|  De aqui EN ADELANTE cualquier cosa que pase es responsabilidad        |
|  vuestra. Protestas dirigirse a /dev/echo o al tlf. 900-666-000        |
```

-----[TABLA DE CONTENIDOS]----
----[SET 29]----

"Este aparato llamado telefono tiene demasiados problemas para ser seriamente considerado como un medio de comunicacion. El aparato no tiene ningun valor inherente para nosotros".

Western Union. Memorandum interno, 1876.

EOF

```
-[ 0x01 ]----------------------------------------------------------------
-[ Editorial ]-----------------------------------------------------------
-[ by SET Ezine ]-------------------------------------------------SET-29--
```

Este es el SET de la treintena. Si, este es SET 29 pero realmente llevamos 30 (Mirar bien los archivos!!!). Afrontamos ya lo que entendemos nuestra "mayoria de edad", en lo que como ezine, SET se refiere, y, aunque a efectos de edicion en este numero no se nota demasiado, espero (de momento solo espero) que sea el ultimo tal y como lo conoceis, tenemos en mente algunos cambios (todos para mejor!!!,... aunque esto es solo nuestra opinion).

Que nadie se alarme, aunque el paso vital a la treintena es sumamente traumatico para estos animales llamados seres humanos, no le va a pasar lo mismo a SET. No habra cambios de concepto, pero es el momento de dar un paso mas hacia adelante (no, no vamos a participar en la serie...), y a efectos de la edicion de SET se notara poco, al menos los proximos numeros.

Somos consciente que despues de esperar seis meses a leer este numero lo ultimo que os apecete es leerse el toston de la edicion, pero quien quiera "sacar nota" ha de pasar por aqui... (os preguntaremos la leccion a todos y los que no supreren la prueba van a tener que repetir).

Por otro lado estamos gratamente sorprendidos del monton de colaboraciones de "esporadicos" que se han animado a enviar articulos (deja de leer ya y ponte a escribir para SET 30!!!) y tambien de la increible disminucion de mensajes de tipo "kiero ser jaquer, henseñame ke kiero", la cuenta pendiente esta en el foro, aunque ya empezamos a ver gente capaz de discutir sin insultar.

... Se estara moviendo algo en la escena?.

No os vamos a hacer un resumen del contenido porque no tenemos nada que resumir, esta bien clarito en el indice...

```
          _ /_ /_ /_ /_ /                           \ _\ _\ _\ _\ _
        _ /_ /_ /_ /_ /   Que los Bits os protejan   \ _\ _\ _\ _\ _
      _ /_ /_ /_ /_ /                                 \ _\ _\ _\ _\ _
    _ /_ /_ /_ /_ /              SET Staff              \ _\ _\ _\ _\ _
  _ /_ /_ /_ /_ /                                       \ _\ _\ _\ _\ _
```

EOF

```
-[ 0x02]----------------------------------------------------------------------
-[ Tutorial de ensamblador AT&T para microprocesadores x86]------------------
-[ by Cemendil ]-----------------------------------------------------SET-29--
       TUTORIAL DE ENSAMBLADOR AT&T PARA MICROPROCESADORES x86
  Autor   :  Cemendil   <cemendil@hotmail.com>
  Fecha   :  29/10/2003
  Version :  0.91
---------------------------| Contenidos. |---------------------------
     Prefacio.
     Introduccion.
     Licencia.
     Herramientas.
     Parte 1 : Ensamblador en formato AT&T.
         1.1 Un codigo de ejemplo.
         1.2 Reglas generales.
         1.3 Algunas directivas interesantes.
           1.3.1. Directivas de almacenamiento de datos.
           1.3.2. Directivas de alineamiento.
           1.3.3. Otras directivas importantes.
         1.4 Usando el preprocesador.
     Parte 2 : Ensamblando subrutinas.
         2.1 Normas de llamada a subrutinas.
            2.1.1. Como llama el compilador de C a una subrutina.
            2.1.2. Como se comporta una subrutina en C.
         2.2 Subrutinas 'canonicas'.
         2.3 Un ejemplo de subrutina 'canonica'.
         2.4 Subrutinas sin 'frame pointer'.
     Parte 3 : Ensamblando 'inline'.
         3.1 El metodo Frankenstein.
            3.1.1 Insercion de ensamblador.
            3.1.2 Injertos monoliticos.
            3.1.3 Un ejemplo retorcido.
         3.2 Introduccion al metodo ortodoxo.
         3.3 Implementando el metodo ortodoxo.
            3.3.1 Simbolos solapados.
            3.3.2 Formatos mas comunes.
            3.3.3 Ejemplos de formatos de la CPU.
            3.3.4 Ejemplos de formatos de la FPU.
            3.3.5 Registros corruptos.
         3.4 Que se nos queda en el tintero.
     Referencias.
----------------------------| Prefacio. |---------------------------
    El presente tutorial explica los fundamentos del ensamblador
en formato AT&T (el mas empleado en maquinas tipo UNIX), para micros
ix86 en modo de 32 bits. Se expondran los fundamentos de este formato
(parte 1), como programar subrutinas compatibles con los compiladores
de C -- especialmente el gcc -- (parte 2), y como intercalar codigo
ensamblador en el interior de un programa en C de manera que gcc sea
capaz de compilarlo de manera optima (parte 3).
    Esta ultima parte es especialmente complicada, puesto que gcc
requiere mucha informacion para poder admitir el ensamblador dentro
del codigo en C. Basta pensarlo un momento: si nuestro fragmento
de ensamblador modifica el registro %ebx, el compilador debe saberlo.
De otro modo podria ocurrir que una variable tipo 'register' almace-
nada en %ebx por el compilador fuese corrompida al ejecutarse el
fragmento de ensamblador, con resultados posiblemente tragicos. Si
por el contrario el compilador esta al tanto, el mismo se encargara
de hacer el baile de registros necesario para que nada se vaya al
garete.
    Para entender este tutorial necesitaras ciertos conocimientos
previos de ensamblador. No es mi intencion hacer un cursillo de
introduccion al ensamblador, si no explicar como sacar partido a
```

la potencia del ensamblado con gcc.
    El que el tutorial tenga mas de 100Kb no debe hacerte pensar que el tema es muy largo o complicado; he escrito este documento con abundancia de ejemplos y procurando ser lo mas redundante posible. A menudo un tema se introduce informalmente, luego se repite de manera formal y finalmente se estudia un ejemplo elaborado. Esto hace que el documento sea muy grande, aunque espero que sea tambien mucho mas instructivo. Realmente es necesario que haya mas gente dispuesta a programar en ensamblador de una manera moderna. Los viejos tiempos, en que los programas se hacian 100% en ensamblador, han pasado, pero eso no quiere decir que tengamos que programar en Visual Basic.
    Creo que un tutorial como este es necesario, dado que es bastante complicado encontrar documentacion en nuestra lengua sobre ensamblado en UNIX. Por mi experiencia en foros, de vez en cuando aparece alguien totalmente confundido por esos garabatos que vomita UNIX cuando las cosas se tuercen. Bueno, si estas en ese grupo, espero que en este articulo encuentres un alivio a tu confusion. Si tienes experiencia en estos temas, quizas las partes avanzadas del tutorial (sobre todo la Parte 3) te sirvan como minimo de referencia.

----------| Introduccion. |----------

    Es muy posible que estes familiarizado con el lenguaje ensamblador para microprocesadores x86, y que hayas escrito algunos programillas usando la notacion Intel, la mas comun para estos menesteres, de la que el siguiente fragmento de codigo es un ejemplo:

```
        or [bx], bp
        add di, ax
        or dh, [bx+si]
        add [301ah], di
        add [si+0ah], bp
        xor al, [bx+di]
        int 3
```

    Lo que haga este codigo es irrelevante. Lo importante es el formato del codigo: estamos ante ensamblador tipo Intel en 16 bits. Este formato es el que obtendras usando la opcion 'u' del debug de MS-DOS, o cualquier desensamblador generico para Windows. Pero si has estado hozando un poco en UNIX, es posible que te hayas encontrado con desensamblados como el siguiente (usa, por ejemplo, 'gcc -S' con cualquier programa en C, y abre el fichero con extension .s que aparece tras la compilacion):

```
        addl $-8,%esp
        movl -8(%ebp),%eax
        pushl %eax
        movzwl -10(%ebp),%eax
        movl 8(%ebp),%edx
        movsbl (%eax,%edx),%eax
        pushl %eax
        call _putc
        addl $16,%esp
```

    Las diferencias respecto al codigo anterior son claras: por un lado, aparecen simbolos curiosos como '%' y '$'. Ademas, instrucciones familiares como 'add' y 'push' adquieren un sufijo, como 'addl' y 'pushl'. Por otro lado, el codigo es de 32 bits, como demuestra el uso de registros extendidos (%eax, %edx, etc.).
    El hecho de que el ensamblador sea de 32 bits no es importante. Naturalmente, puedes programar en 32 bits con formato Intel, por ejemplo con nasm. De lo que nos ocuparemos sera de explicar las primeras diferencias, las referentes a la notacion.
    Algunos hackers de bastante prestigio detestan el formato AT&T: basta ver la muestra de arriba para darse cuenta de que hay que escribir mucho mas que con formato Intel, con todos

esos ’%’, ’$’ y sufijos. Sin embargo, la cosa es como el eterno debate entre vi y emacs: puede que el formato AT&T sea poco amigable, pero si te acostumbras acaba siendo casi irreemplazable. Yo emepece programando en formato Intel, pero una vez que me hice a programar a la AT&T, ya casi no puedo ver el ensamblado en modo Intel. Como no es nuestra tarea iniciar ’holy wars’ por motivos tan irrelevantes (aunque nos encante ;) sera mejor que juzgues por ti mismo. En ultima instancia, el formato AT&T es inevitable en ciertas ocasiones, por ejemplo al usar gdb.

Si no tienes experiencia en el tema tratado por este tutorial, mi consejo es que lo leas con una consola abierta, dispuesto a compilar y experimentar con cada ejemplo que se propone. Es dificil captar muchos aspectos de estas cuestiones sin dedicar un tiempo a la practica.

Un documento tan largo, y que pretende cubrir un area tan extensa, necesariamente tiene deficiencias. Si observas alguna, por favor enviame un mail. Tambien puedes editar el documento libremente (consulta la licencia).

----------| Licencia. |----------

Eres invitado a copiar, distribuir y modificar libremente este documento, con las siguientes restricciones:

i) En caso de modificacion: o bien eliminaras toda referencia a mi (Cemendil) del documento, o bien indicaras en lugar bien visible quien eres y que cambios has introducido.

ii) El autor no se hace responsable de cualquier contingencia derivada del uso, correcto o incorrecto, de este documento.

----------| Herramientas. |----------

Lo unico necesario para ensamblar a gusto en el formato AT&T es el ensmablador GNU, el gas. Este viene con todas las distribuciones de UNIX libres, de modo que no tendras problemas en este sentido. Para moverte en las partes 2 y 3 necesitaras tambien el compilador gcc; de nuevo, en UNIX no hay problema.

Para conseguirte las ultimas versiones de gas o gcc, visita www.gnu.org y dirigete a un mirror ftp que contenga las distribuciones en codigo fuente (ftp.aeneas.mit.edu, en /pub/gnu/, por ejemplo). Para conseguir gas, descargate las ’binutils’. La ultima version en fecha de escribir esto es la 2.6. Para conseguir ’gcc’, vete al directorio de ese nombre. La ultima version disponible es la 3.3. Ten en cuenta que si quieres montarte toda la suite de compilacion tambien necesitaras el linkador ’ld’.

Ahora, si tienes Windows tambien puedes disfrutar del ensamblado en este formato. Para ello, consiguete el gcc para Windows. Yo conozco dos distribuciones, el djgpp (que mas bien es para MS-DOS, al menos la ultima vez que lo use), y el Bloodshed Dev-C++, que tiene una GUI incorporada y es bastante agradable de usar. Cualquiera de ellos te lo puedes conseguir navegando un poco por la red. Ten en cuenta que el Visual C de Microsoft trabaja solo en formato Intel. De todos modos, el Visual C cuesta un ojo de la cara, asi que si no quieres caer en los oscuros abismos de la pirateria, es bueno saber que tienes esas alternativas que, si no son tan versatiles como el VC, al menos te permiten salvar la candidez de tu alma :).

Hace bastante tiempo vi por la red un programa llamado ’gas2masm’; el programa traducia automaticamente entre ensamblador Intel y AT&T. Si no recuerdo mal, venia una copia con el codigo fuente del Quake. Una herramienta semejante, que debe andar por algun lado en la red, puede resultarte muy util para hacer comparaciones. El programa es sencillo, de modo que puedes imaginarte que las diferencias entre los dos formatos son bastante faciles de aprender.

----------| PARTE 1 : Ensamblador en formato AT&T |----------

En esta parte nos ocuparemos de la estructura del ensamblador en formato AT&T, y de como programar con gas. Ten en cuenta que esto es un tutorial, no un 'white paper'. Los conceptos del ensamblador estaran mezclados con el uso de gas: por ejemplo, '.p2align' es una directiva de gas que no tiene por que ser reconocida por cualquier otro ensamblador en formato AT&T. En fin, dominando gas y usando un poco el sentido comun es posible adaptarse a cualquier otro ensamblador de ese mismo genero (en el dudoso caso de que te encuentres con uno que no sea gas).

---| 1.1 Un codigo de ejemplo.

Lo mejor para familiarizarse con la situacion es trabajar sobre un ejemplo que podamos comprender. Nuestro ejemplo sera la siguiente subrutina (no empleamos un programa principal para ahorrarnos, por ahora, problemas con los valores pasados y devueltos):

```
/* Subrutina hola mundo. */
#include <stdio.h>
static char msg[] = "Hola, mundo!";
void saluda(void)
{
     register short i = 0;
     while (++i < 10)
         if (i == 5) puts(msg);
}
```

Supongamos que tenemos este codigo fuente en el fichero 'hola.c'. En estas condiciones, solo tenemos que hacer desde una shell:

```
demeter# gcc -S hola.c
demeter# ls
hola.c  hola.s
```

Si ahora editamos hola.s, obtenemos:

```
        .file    "test.c"
gcc2_compiled.:
___gnu_compiled_c:
.data
_msg:
      .ascii "Hola, mundo!\0"
.text
      .align 4
.globl _saluda
      .def  _saluda;  .scl 2;  .type 32;  .endef
_saluda:
      pushl %ebp
      movl %esp,%ebp
      subl $20,%esp
      pushl %ebx
      xorl %ebx,%ebx
     .p2align 4,,7
L3:
      incw %bx
      cmpw $9,%bx
      jle L5
      jmp L4
      .p2align 4,,7
L5:
      cmpw $5,%bx
      jne L6
      addl $-12,%esp
        pushl $_msg
      call _puts
      addl $16,%esp
L6:
      jmp L3
```

```
              .p2align 4,,7
        L4:
        L2:
              movl -24(%ebp),%ebx
              leave
              ret
             .def  _puts;  .scl 2;  .type 32;  .endef
```

Bien, aunque esto parece muy complicado, en realidad tenemos tan solo cuatro tipos de comandos genericos:

1) Directivas como .p2align , .align , .file , .text, etc.
2) Etiquetas como L2: , _msg: , _saluda: , etc.
3) Instrucciones en ensamblador (el resto).
4) Comentarios, que aqui no aparecen. Los comentarios son identicos a los de C, es decir, van entre '/*' y '*/'.

Podemos ver que todos los comandos estan bien delimitados. Las directivas empiezan con un punto, las declaraciones de etiquetas acaban con dos puntos, los comentarios van como en C, y el resto son instrucciones de ensamblador. (Aveces, algunos compiladores ponen un punto antes de las etiquetas, p. ej. '.L0:'. No confundas esto con una directiva).

Por directivas entendemos instrucciones especiales que indican al ensamblador que hacer en ciertas situaciones. Por ejemplo, las directivas .data y .text le indican al ensamblador donde estan los datos y donde esta el codigo. Las directivas tipo .align le dicen al ensamblador que alinee el codigo o datos respecto a una cierta cantidad de bytes (lo tipico es 4, una doble palabra, pero para ciertos micros el optimo puede ser hasta 16, dependiendo de la situacion). La directiva .globl indica al compilador que 'saluda' es el nombre de la subrutina a efectos de linkado. La directiva .ascii marca el inicio de una cadena de caracteres. De la misma manera, una directiva .byte indicaria una cadena de bytes. Existen otras directivas de las que nos ocuparemos en la seccion 1.3.

Nota: ten en cuenta que ciertas directivas solo las emplea en la practica el compilador. No es necesario aprender todas las directivas, si no las mas relevantes. Un buen ejemplo es la directiva .def, que introduce informacion para el debugger.

Entender las etiquetas es mas facil. Podemos ver como los saltos (jmp, jle) se refieren claramente a las etiquetas. Por otro lado, tenemos una instruccion mas que se ocupa de etiquetas. Es la 'pushl $_msg'. Como es facil suponer, lo que hace esta instruccion es meter en la pila la direccion de la cadena asociada a la etiqueta '_msg:', es decir, un puntero a "hola, mundo!\0". Analicemos un poco la instruccion, dado que es un buen ejemplo del formato. Tenemos:

```
        pushl $_msg
```

Los elementos raros aqui son el sufijo 'l' a push y el '$' antes de la etiqueta. Es importante que introduzcamos ahora dos nociones fundamentales:

A) En el formato AT&T, toda instruccion que realiza una modificacion de datos _debe_ indicar la magnitud de los datos modificados. Cada magnitud se indica con un sufijo a la instruccion, y hay en total tres magnitudes: byte (sufijo 'b'), palabra (sufijo 'w') y doble palabra (sufijo 'l'). El byte son 8 bits, la palabra 16, y la doble palabra 32. En general, se corresponden con los tipos enteros en C: char, short, long.

Ejemplos (no todos los ejemplos estan en el codigo de arriba):

```
1) pushl $_msg   : Dado que un puntero es de 32 bits,
                   'push' lleva el sufijo 'l'.
2) cmpw $5, %bx  : Dado que %bx es un registro de 16
                   bits, 'cmp' lleva el sufijo 'w'.
```

```
      3) inb $20, %al  : Para el registro de 8 bits %al, la
                         instruccion 'in' lleva el sufijo 'b'.
      4) ret           : Esta instruccion no modifica datos, asi
                         que no lleva sufijo. (Ya se, en rigor
                         modifica %eip, pero es que _todas_ las
                         instrucciones modifican %eip, no?)
NOTA: Para la programacion de instrucciones en coma flotante y
      MMX aparecen otros sufijos, o los mismos con diferente
      significado. Si vas a programar codigo con la FPU o MMX,
      deberias leerte la documentacion de gas, donde se detallan
      todos los sufijos. Algunos ejemplos de la parte 3 de este
      tutorial emplean la FPU; puedes consultarlos para orientarte.
    B) Cuando se quiere indicar una cantidad 'inmediata' en un
movimiento de datos, siempre se indica el prefijo '$' antes de
esa cantidad. Igualmente, cuando un '$' precede a una etiqueta,
se hace referencia a la direccion de memoria a la que apunta la
etiqueta. Si no le precede un '$', se entiende que se hace
referencia al contenido de la posicion apuntada por la etiqueta.
(Aqui el '$' juega un papel analogo al del '&' en C: sin '&' se
da el contenido, con '&' se da la direccion).
    Ejemplos:
    1) pushl $_msg   : Mete el puntero de 32 bits que apunta a
                       "Hola, mundo!\0" en la pila.

    2) pushb $0x5a   : Mete en la pila el byte 0x5a.
    3) pushw $0xbb5a : Mete en la pila la palabra 0xbb5a.
    4) pushl _msg    : Mete en la pila la dpalabra formada por los
                       caracteres 'H', 'o', 'l' y 'a'.
    Observa como los numeros hexadecimales se indican exactamente
igual que en C. Por defecto, si un numero no lleva el prefijo '0x',
se asume que es decimal. Exactamente igual que en C.
ADVERTENCIA: Si una cantidad inmediata no va precedida por un
===========  '$', el ensamblador entiende que se trata de una
             referencia a la memoria.
      Ejemplo:
         pushl 0x5a  : Mete en la pila la doble palabra que se
                       encuentra en la direccion de memoria 0x5a
                       en el segmento de datos.
      Ten muy en cuenta esta advertencia, dado que el olvidarse de
      un '$' es un motivo tipico para obtener un desesperante
      'segmentation fault - core dumped' tras otro, al acceder el
      programa a la memoria arbitrariamente en vez de almacenar
      datos inmediatos.
      Salvo que estes programando un sistema operativo, un sector de
      arranque o algo asi, nunca querras direccionar memoria directa-
      mente, de manera que sospecha de toda cantidad inmediata que
      vaya sin su correspondiente '$'.
    Bueno, una vez que tenemos cierta idea de las directivas,
etiquetas y que quieren decir los '$', vamos a ver que quieren
decir los '%'. Un simple vistazo nos muestra que siempre que
aparece un '%' es como prefijo a un nombre de registro: %ebp,
%ebx, %bx, etc. Precisamente la funcion del signo '%' es la de
hacer que los nombres de los registros no se puedan confundir
con el de alguna variable o subrutina escrita en C. Ten en cuenta
que gas es el ensamblador que usa gcc, asi que si en un programa
en C usas una variable a la que llamas 'eax', o una subrutina 'esp'
(por 'espera', por ejemplo) el ensamblador debe saber que no te
estas refiriendo a un registro. Obviamente, no es tarea del gcc
anunciar al programador que esta usando nombres de registros: se
supone que C es un lenguaje de alto nivel. Asi pues, al ensamblar,
antes de un nombre de registro hay que escribir siempre el signo
'%'. Una vez que te acostumbras a ello, es una ayuda el poder
```

```
diferenciar de un vistazo donde se usan variables y donde registros.
NOTA: De todas maneras, algunas versiones de 'gcc' tienen otro
      mecanismo de seguridad para no confundir simbolos del
      ensamblador con simbolos del codigo en C. El mecanismo
      consiste en prefijar todo nombre declarado en C con un
      subrayado '_'. Si observas el codigo mas arriba, podras
      ver como la variable 'msg' es ensamblada como '_msg', y
      la subrutina 'saluda' es asociada a la etiqueta '_saluda'.
      Este mecanismo de seguridad se explicara con mas detalle
      en la seccion 2.1.2. Ten en cuenta que no todas las
      versiones de gcc implementan este mecanismo.
    Con esto tenemos casi todo lo relativo al formato AT&T. Bueno,
falta un par de cosillas... que suelen disuadir a mucha gente de
usar este formato. Observa esta linea, al principio del codigo:
        subl $20, %esp
    Bien, ahora podemos entender la cosa facilmente. Es una
instruccion 'sub' (restar), que actua sobre magnitudes de
32 bits (sufijo 'l'), y se refiere a una cantidad inmediata
de valor 20 decimal ($20), y al registro esp (%esp). Pero
un momento ... segun el convenio Intel, lo que haria esta
instruccion seria restar a una cantidad inmediata el contenido
de un registro! En otras palabras, equivaldria a 20 = 20 - %esp
en pseudocodigo, lo que no tiene sentido.
    Lo que sucede, naturalmente, es que en formato AT&T el
orden de los operadores va _al_reves_ que en el formato Intel.
Si en el formato Intel tenemos la forma general:
     instruccion  destino, fuente   ( p. ej.  sub esp, 20 )
en el formato AT&T tenemos la forma general:
     instruccion  fuente, destino   ( p. ej. subl $20, %esp )
Esto echa para atras a montones de gente, que de pronto ven que
tienen que 'volverse zurdos' (o diestros ;) para manejar este
nuevo formato. Mi consejo para cambiar rapido el chip es el
siguiente: IMAGINATE QUE ESTAS LEYENDO EL CODIGO. Si, como si
estuviese escrito en español o ingles. Veamos: como escribirias
en una frase lo que hace la instruccion 'subl $20, %esp'. Pues
mas o menos: "resta 20 a esp". Si te fijas, el orden de los
operadores va exactamente igual que en el formato AT&T.
    Asi, las lineas anteriores a la que hemos visto son:
       pushl %ebp              " %ebp a la pila    "
       movl %esp,%ebp          " mete %esp en %ebp "
       subl $20,%esp           " resta $20 a %esp  "
             ...                     ...
Vaya, no se lo que pensaras tu, pero una vez que te acostumbras
la cosa es bastante natural. Lo unico que tienes que hacer es
leer de izquierda a derecha.
<holy_wars>
Compara eso con 'sub esp, 20'. Para entender esta instruccion
debes leerte el mnemonico 'sub' en la izquierda, luego irte al
extremo derecho para ver la fuente y entonces al centro para
ver el destino. En esta instruccion quizas no sea muy molesto,
pero hay codigos con instrucciones mas cargadas que una pizza
especial. Ademas, leer estas instrucciones es como leer
"restale a esp 20", lo cual es sintacticamente correcto, pero a mi
me parece definitivamente incomodo. Para que la notacion Intel
tuviera sentido deberia ser, creo yo, 'esp sub 20', que se
podria traducir por "esp menos 20", lo cual tiene mas sentido.
</holy_wars>
    Con tener en cuenta esto, ya puedes leer practicamente todo
el listado en ensamblador del ejemplo, con una excepcion. La linea
        movl -24(%ebp), %ebx
    Esta es la ultima peculiaridad del formato AT&T. La expresion
(%ebp) indica al ensamblador que se emplee la direccion de memoria
```

```
a la que apunta %ebp; en esencia, indica que %ebp actua como
puntero. El -24 es un desplazamiento respecto a ese puntero. La
expresion -24(%ebp) indica que la fuente del 'movl' es la doble
palabra almacenada en la direccion 24 bytes por debajo de la
apuntada por %ebp.
    En su forma mas general, el direccionamiento en un i386 tiene
un puntero base, un desplazamiento inmediato, un indice y un factor
de desplazamiento del indice. Por ejemplo,
        movl -120(%eax, %ebx, 4), %ecx
almacena en %ecx el contenido de la direccion de memoria indicado
por 120 + %eax + 4*%ebx. La forma general del direccionamiento es
por tanto:  desp(base, ind , fact) .
        Ejemplos:
        AT&T                    Intel
        (%eax)                  [eax]
      -5(%eax)                  [eax-5]
        (%eax,%ebx)             [eax + ebx]
   0x543(%ebp,%eax,2)           [ebp + eax*2 + 543h]
      13(,%eax,2)               [eax*2 + 13]
    Lo interesante de esta notacion de direccionamiento es que
es muy homogenea. Cada cosa esta en su lugar, y es facil de
localizar. Hacer un direccionamiento es como llenar una ficha.
NOTA: Adicionalmente, puedes indicar un registro de segmento
      respecto al cual direccionar la memoria, por ejemplo,
      %ds:20(%esi, %ebx, 2). Sin embargo, en la practica es
      muy poco comun usar esta opcion.
    Ahora si que tenemos todo lo necesario para entender el
codigo en ensamblador que proponiamos al principio. Vamos a
ver algunas partes concretas.
    Salvando las directivas iniciales, las primeras lineas son:
              pushl %ebp
              movl %esp,%ebp
              subl $20,%esp
              pushl %ebx
estas lineas son muy comunes en subrutinas. Lo que se esta
haciendo es conservar la referencia de pila (%ebp) de la subrutina
que nos llama (pushl %ebp), definir una nueva referencia
(movl %esp,%ebp), definir una total de 20 bytes para almacenamiento
de variables locales (subl $20, %esp) y finalmente se conserva
el valor de %ebx en la pila (pushl %ebx), dado que se va a usar
como variable registro. Veremos mas de esto en la Parte 2.
NOTA: El compilador gcc intenta por todos los medios mantener
      la pila alineada en 16 bytes al comienzo de las variables
      locales de cada subrutina, por motivos de eficiencia. Por
      eso se reservan en la pila 20 bytes que no se van a emplear
      para nada. Observa tambien los ajustes en la pila antes
      y despues de la llamada a puts.
    Ahora vamos con el cuerpo del programa:

        L3:
              incw %bx
              cmpw $9,%bx
              jle L5
              jmp L4
             .p2align 4,,7
        L5:
              cmpw $5,%bx
              jne L6
              addl $-12,%esp
              pushl $_msg
              call _puts
              addl $16,%esp
```

```
        L6:
                jmp L3
                .p2align 4,,7
        L4:
        L2:
```

Una primera observacion es que gcc alinea siempre los saltos (directiva '.p2align', que comentaremos en la seccion 1.3.2), lo cual es una politica muy sensata. Los saltos mal alineados son tremendamente ineficientes.

Este codigo tiene un fallo: hay muchos mas saltos de los necesarios, pero compilando con -O la cosa cambia mucho (haz la prueba). Por lo demas, es sencillo reconocer lo que pasa: desde L3 a L5 tienes la condicion del bucle, observa que %bx se corresponde con el entero corto 'i' del programa en C. Naturalmente, es tratado con el sufijo 'w'. Desde L5 a L6 tenemos el if, incluyendo la llamada a puts. De L6 a L4 se cierra el bucle.

Finalmente queda por ver la salida de la subrutina:

```
             movl -24(%ebp),%ebx
             leave
             ret
```

el par leave, ret es el usual en estos casos. El movl recupera el valor de %ebx que se metio en la pila al comienzo de la subrutina. Lo mismo habria dado hacer popl %ebx.

---| 1.2 Reglas generales.

Pasemos a resumir todas las reglas que hemos observado, para tener una referencia rapida.

REGLAS FUNDAMENTALES:

1) El orden de los operadores es:

```
    OPERACION  FUENTE, DESTINO     ( ej. adcl $26140, %eax )
```

2) La magnitud de los operadores esta determinada por el sufijo del 'opcode'. Los sufijos de la ALU son 'b', 'w', 'l', que corresponden respectivamente a 8, 16 y 32 bits.

```
     (ej. incb %al  ;  pushw $0x5a  ;  xorl %eax, %eax )
```

NOTA: gas es capaz de interpretar correctamente las instrucciones que no lleven sufijo pero cuya magnitud sea obvia, como 'inc %al', pero dara error cuando no sea asi, como en 'push $0x5a'.

3) Cantidades inmediatas y referencias.

3.a) Las cantidades inmediatas deben ir precedidas de un '$', o se consideraran referencias absolutas de memoria.

```
     ( movl $0x5abb, %eax  !=  movl 0x5abb, %eax )
```

3.b) Si una etiqueta _no_ va precedida de un '$' se esta haciendo referencia al contenido de la zona de memoria a la que apunta la etiqueta.

```
     (ej. movl variable, %eax   /* Almacena contenido de */ )
     (                          /* 'variable' en %eax    */ )
```

Si escribes un '$' antes de la etiqueta, se hace referencia a la etiqueta como puntero.

```
     (ej. movl $variable, %eax  /* Estas instrucciones   */ )
     (    movl %eax,  ptr       /* equivalen en C a:     */ )
     (                          /* ptr = &variable;      */ )
```

4) La forma general de direccionamiento es:

```
          SEG:DESP(BASE,IND,FACT)
```

donde:

```
     SEG  == registro de segmento a emplear.
     DESP == desplazamiento inmediato a usar.
     BASE == registro base.
     IND  == registro indice.
     FACT == factor de multiplicacion del indice.
```

por ejemplo, leal 120(%eax, %eax, 2), %eax equivale a %eax = 120 + %eax + 2*%eax en pseudocodigo.

```
    5) Los nombres de registros van siempre precedidos de '%'.
        ( ej. %aex, %ebx, %ecx, %edx, %esi, %edi, %esp, %ebp )
  UNA CONVENCION IMPORTANTE:

    En el codigo de la seccion 1.1 habras observado que los
nombres de las etiquetas de los bucles son L2, L3, L4, etc.
En cualquier programa en ensamblador existen etiquetas que
no quieres que sean visibles para el linkador; un caso
especialmente importante es el de las etiquetas contenidas
en las macros.
    Cualquier etiqueta que comience por 'L' (ele mayuscula),
es considerada por gas como una _etiqueta_local_ y por tanto
es traducida directamente  a una direccion en memoria sin dejar
ninguna referencia en el fichero objeto generado.
    Ejemplos de etiquetas locales:
        L0, L1, ... , L9    (estas son especiales, ver mas abajo)
        Lsalto, Lfalsa, Lloop7, ...
    Las etiquetas L0, ... , L9 son especiales porque puedes
redefinirlas tantas veces como quieras (gas se encarga de
interpretar a cual te refieres en cada momento). Esto las hace
especialmente bien dotadas para las macros (tienes un ejemplo
de este uso en la seccion 1.4, fichero 'arch.h'). Por ejemplo,
puedes definir L0 noventa veces en el mismo programa. En esos casos,
si quieres referirte a la etiqueta L0 inmediatamente anterior usa
'L0b' (L0 'backwards'), y si quieres referirte a la inmediatamente
posterior usa 'L0f' (L0 'forward').
    Por lo tanto, puedes considerar a las etiquetas locales como
'etiquetas basura', que usas en una parte del programa y a
continuacion olvidas.
    Para mas datos sobre el uso de las L0...L9 consulta la
documentacion de gas, donde se precisa como se almacenan
internamente.
    Una detalle que merece explicacion: en la siguiente seccion
veremos que para exportar al linkador un simbolo es necesario
usar la directiva .globl (o .global). Entonces, dado que
cualquier etiqueta no deberia ser exportada a menos que se
use .global, ?por que molestarse en usar etiquetas locales?
Bueno, la realidad es que aunque no exportes una etiqueta,
cierta informacion de la misma si se exporta, y eso causa que
'gdb' se haga un lio a la hora de desensamblar las subrutinas
que has escrito. Una etiqueta no local pero no declarada .global
no es lo bastante global como para llamarla desde el linkador,
pero es lo suficientemente no local como para que el debugger
la trate como una subrutina aparte. Mi consejo es que emplees
etiquetas locales dentro de lo posible, y que recurras a las no
locales solo en situaciones importantes. Si esto te parece un
lio, ten en cuenta que el uso de etiquetas no locales no afecta
para nada a la eficiencia del codigo; puedes pasar completamente
de las etiquetas locales.
    En la seccion 1.4 tienes un ejemplo en el que 'gdb' es
capaz de desensamblar el codigo asociado a una etiqueta no
local que sin embargo no ha sido declarada .global. Aunque te
parezca una ventaja poder desensamblar el codigo en cada
etiqueta que usas, esto tiene un inconveniente: 'gdb' solo
desensamblara de una vez el codigo que haya entre dos etiquetas
no locales dadas. Esto puede resultar incomodo si abusas de las
etiquetas no locales.
---| 1.3 Algunas directivas interesantes.
    Por lo general, un programa en ensamblador sin directivas es
completamente inutil. Veamos algunas directivas interesantes; la
referencia completa la puedes encontrar en la documentacion de gas.
   1.3.1. Directivas de almacenamiento de datos.
```

```
     .ascii : almacena en memoria una o varias cadenas de
              caracteres. Ten en cuenta que esta cadena no
              es null-terminada (cf. la directiva .asciz)
         (ej:  .ascii "Hola, mundo!", "\n\0" )
     .asciz : identico a .ascii, pero se incluye '\0' al final
              de cada cadena.
     .byte  : almacena en esa posicion de memoria uno o mas
              bytes. Esta instruccion es interesante para
              definir instrucciones no soportadas por el
              ensamblador.
         (ej: .byte 0x0f, 0x31        /* Incluir estos datos en */ )
         (                            /* el codigo equivale a   */ )
         (                            /* la instruccion  rdtsc  */ )

     .double : almacena una o varias cantidades en coma flotante,
               de doble precision (64 bits).
         (ej: .double 0.314159265358979324e+1 )
     .float  : almacena una o varias cantidades en coma flotante,
               de precision simple (32 bits). (Sinonimo de .single)
         (ej: .float 2.718281, 0.69314718055 )
     .hword  : almacena una o varias palabras de 16 bits. Es
               sinonimo de .short.
     .int    : almacena una o varias dpalabras (32 bits). Es
               sinonimos de .long.
     .octa   : almacena uno o varios numeros de 16 bytes (128 bits)
               de longitud.
     .quad   : almacena una o varias cuadruples palabras (64 bits
               de longitud). Puede ser muy util.
                  (ej: .quad 0x000000003040ffff )
   1.3.2. Directivas de alineamiento.
     .align  : alinea la siguiente etiqueta/instruccion en la
               cantidad de bytes indicada. Esta directiva puede
               tomar hasta tres parametros:
               .align AL, PAD, SKIP
                   AL   : bytes respecto a los que alinear.
                   PAD  : byte con el que rellenar.
                   SKIP : la directiva sera ignorada si hay que
                          añadir mas de SKIP bytes para alinear.
     (ej:  .align 16,0,7  /* Alinea por 16 bytes a cero, pero */ )
     (                    /* que eso no suponga rellenar mas  */ )
     (                    /* de 7 posiciones.                 */ )
                Si en esta directiva no indicas el campo PAD,
                el ensamblador rellena con instrucciones tipo
                NOP, lo cual es conveniente en el codigo, claro.

     .p2align : identico a .align, pero el alineamiento se hace
                segun el numero de bits indicado por el primer
                parametro de la directiva. Asi, .p2align 2
                equivale a .align 4 y .p2align 4 equivale a
                .align 16. Para ejemplos del uso de .p2align,
                observa el codigo de la seccion 1.1.
   Existen otras directivas de alineamiento; si te interesa
emplearlas, consulta la documentacion de gas.
    1.3.3. Otras directivas importantes.
      .data    : indica a gas que los siguientes comandos deben
                 ensamblarse en la seccion de datos. Esta directiva
                 admite un comando, el numero de seccion, pero es
                 dudoso que lo encuentres util. Consulta la
                 documentacion, si crees que te puede servir de algo.
      .equ     : un clasico de los ensambladores. Iguala un simbolo
                 a una expresion, como '.equ STDIN 0'.
      .equiv   : identico a .equ, pero si el simbolo ya ha sido
```

```
                definido, gas aborta con un error. Puede resultar
                util.
        .file    : indica a gas que el codigo que sigue corresponde
                a cierto fichero. Puedes ver un ejemplo en el
                codigo de la seccion 1.1.
        .global  : declara cierto simbolo como global, de manera que
                'ld' y 'gdb' pueden reconocerlo. Esto es fundamental
                a la hora de programar subrutinas, y puede ser muy
                util para definir breakpoints particulares para
                'gdb'.
            (ej:  .global subrut           /* Hace visible para */ )
            (     subrut:                  /* 'ld' el nombre    */ )
            (     ...                      /* 'subrut'.         */ )
        .globl    : sinonimo de .global.
        .include : permite cargar un fichero dentro de otro. Esto
                puede emplearse, por ejemplo, para cargar cabeceras
                o subrutinas interesantes dentro de un programa.
                  (ej:  .include "comun.s" )
        .rept    : repite una cierta seccion de codigo tantas veces
                como las especificadas en el parametro. El final
                de la zona a repetir se indica con '.endr'.
                          (ej:  .rept 10       )
                          (         movsd      )
                          (     .endr          )
        .text    : indica a gas que debe ensamblar los siguientes
                comandos en la seccion de codigo. En realidad,
                .data y .text significan algo mas generico que
                'datos' y 'codigo', pero esta distincion funciona
                perfectamente en todos los casos.
```

Existen muchas otras directivas, algunas de las cuales pueden resultarte de utilidad en ciertos casos. Recuerda que una de las mayores ventajas de gas es que si necesitas un ejemplo de como codificar algo, siempre puedes recurrir a un programa en C que haga algo parecido, y compilar con -S. Esta tecnica te proveera de infinidad de ejemplos interesantes que funcionan de verdad.

Unas directivas interesantes son las dedicadas a definir macros. Las he omitido del documento por dos motivos: en primer lugar, es algo pesado explicarlas (consulta la documentacion si realmente estas interesado; no es complicado). En segundo lugar, el ensamblador ya es bastante ilegible sin necesidad de macros. Francamente, si la cosa esta tan mal, podrias considerar el uso de un lenguaje de alto nivel, y afinar las zonas criticas en vez de escribir un tochoprograma en ensamblador. De todos modos, en la siguiente seccion explicaremos como usar el preprocesador de gcc en el codigo ensamblado, lo que permite usar un monton de facilidades de gcc (#define, #ifdef, etc.) dentro de tu programa ensamblado, lo que hace la vida mas facil y ahorra la necesidad de macros en bastantes casos. La idea es usar la mayor cantidad de mecanismos conocidos antes que aprender otros nuevos. ?Pereza mental? Yo prefiero llamarlo economia.

---| 1.4 Usando el preprocesador.

Una de las mejores caracteristicas de gas es su compenetracion con gcc. El primer ejemplo que veremos, antes del plato fuerte de la parte tercera, sera el uso del preprocesador.

En primer lugar, recuerda que los ficheros que continen codigo ensamblador AT&T siempre tienen la extension '.s'. A los ficheros que contengan ensamblador AT&T que requieren preprocesamiento se les pone siempre la extension '.S'.

Ejemplo: hola.s <-- Fichero con codigo ensamblador normal.
          macro.S <-- Fichero ensamblador con preprocesamiento.

Si quieres algun ejemplo de fichero .S, echale un vistazo al codigo fuente del nucleo de cualquier UNIX. Podras ver una alegre

mezcla de ensamblador y directivas del 'cpp'.
    Por ejemplo, al observar el codigo fuente de mi FreeBSD, localizo rapidamente varios ficheros .S en /sys/i386/boot/biosboot. Estos ficheros incluyen un fichero de cabecera llamado 'asm.h' empleando #include "asm.h", una instruccion del preprocesador. Este asm.h contiene varias macros bastante utiles, por ejemplo:

```
              ( fragmento de asm.h de FreeBSD )
        ...
        #define String       .ascii
        #define Value        .word
        #define Times(a,b)   (a*b)
        #define Divide(a,b)  (a/b)
        #define INB          inb    %dx, %al
        #define OUTB         outb   %al, %dx
        #define INL          inl    %dx, %eax
        #define OUTL         outl   %eax, %dx
        ...
```

    La ventaja de usar el preprocesador es evidente. Otro ejemplo mas mundano de ensamblador con preprocesamiento podria ser el siguiente (hazlo tu mismo):

    1) Crea un fichero 'arch.h' con las siguientes lineas:

```
       #ifdef i686
       #define ALGN 16,,8
       #define IFR(C1, C2, FROM, TO)   cmpl C1, C2      ; \
                                       cmovel FROM, TO
       #else /* i686 */
       #define ALGN 4
       #define IFR(C1, C2, FROM, TO)   cmpl C1, C2      ; \
                                       jnz L8f          ; \
                                       movl FROM, TO    ; \
                                       .align 4         ; \
                                       L8:
       #endif /* i686 */
```

NOTA: En gas, el separador de varias instrucciones dentro de la misma linea es el simbolo ';', exactamente igual que en C. Mira las macros de arriba: cada vez que hay que separar dos instrucciones en la misma linea empleamos el ';'. Aunque el preprocesador expanda la macro IFR en una sola linea, gas sera perfectamente capaz de diferenciar los comandos individuales.

    2) Ahora crea un fichero 'dumb.S' con las siguientes lineas:

```
       #define i686
       #include "arch.h"

           .file "dumb.S"
           .text
       nada:
           .align ALGN      /* Alineamiento dado por arch.h */
           nop
           IFR(-5(%edx), %eax, %esi, %ebx)
           nop
```

    3) Desde la shell, ejecuta el preprocesador:

```
       demeter# cpp dumb.S > dumb.i

       demeter# ls
       dumb.i dumb.S
```

    4) Edita dumb.i, y obtendras:

```
        ( algunos datos del preprocesador al principio )
         .file "dumb.S"
         .text
```

```
        nada:
           .align 16,,8
           nop
           cmpl  -5(%edx),   %eax    ; cmovel  %esi,   %ebx
           nop
```

Este programa en ensamblador no hace nada (util), pero es un buen ejemplo de preprocesamiento.

Ahora veamos como se compila un fichero '.S'. No puede ser mas sencillo. Para compilar 'dumb.S' simplemente debes recurrir a gcc, como si de un programa en C se tratara. Dado que nuestro programa en ensamblador no es un programa principal (le falta el _main), lo compilaremos como un modulo:

```
         demeter# gcc -c dumb.S
         demeter# ls
         dumb.S  dumb.i  dumb.o
```

Ya que hemos llegado hasta aqui, confirmemos que esta todo bien compilado:

```
         demeter# gdb dumb.o
         (gdb saluda y suelta la retahila)
         (gdb) disassemble nada
         0x0 <nada>:     nop
         0x1 <nada+1>:   cmp 0xfffffffb(%edx), %eax
         0x4 <nada+4>:   cmove %esi,%ebx
         0x7 <nada+7>:   nop
         0xa <nada+10>:  add %al,(%eax)
         ...
         End of assembler dump.
         (gdb)
```

Bueno, aqui esta nuestro programa, bien compilado. Recuerda lo que indicamos al final de la seccion 1.2: aunque no hemos declarado como .global a la etiqueta 'nada', el gdb la reconoce perfectamente. Esto sucedera con todas las etiquetas no locales que empleemos. De paso, esta version de 'gdb' tiene la mala costumbre de comerse los sufijos de magnitud cuando no son necesarios.

Resumiendo, puedes usar 'gcc' para compilar un programa '.S' exactamente de la misma manera que con un programa en C. Esto ayuda bastante a la hora de definir makefiles, aunque por lo general deberas separar los ficheros '.S' de los '.c' en el makefile, puesto que gcc suele llevar opciones de compilacion diferentes para ambos. Hacer 'gcc -O3' a un fichero en ensamblador no es incorrecto, pero no tendra ningun efecto y es inelegante.

----------------| PARTE 2 : Ensamblando subrutinas |----------------

La manera mas sencilla de programar en ensamblador consiste en hacer el esqueleto del programa en C y desarrollar las subrutinas mas importantes en ensamblador. Esta estrategia es muy eficente, dado que descarga todo el trabajo pesado en el compilador, y todo el trabajo delicado en el programador.

Sin embargo, existe un desventaja. Si la subrutina es muy corta, la perdida de tiempo asociada a las manipulaciones de pila y los saltos ('call' y 'ret') pueden contrarrestar las ventajas de usar ensamblador. En estos casos es mejor emplear el ensamblado 'inline' (ver la parte 3).

Estudiaremos dos modos fundamentales de programar subrutinas. El modo 'canonico', que consiste en emplear el puntero base de pila %ebp para referirse a las variables locales. Cualquier subrutina compilada con 'gcc -S' usara este modo. El modo 'no canonico' consiste en emplear el puntero de pila %esp como referencia para las variables locales, lo que nos permite ratear el registro %ebp para nuestro propio uso. Puedes generar ejemplos de este tipo a partir de ficheros '.c' compilando con

'gcc -fomit-frame-pointer -S'.
---| 2.1 Normas de llamada a subrutinas.

Para entender lo que sigue, debes de tener bien claro lo que es la pila de tu microprocesador, y como funciona. Debes saber como se altera la pila al hacer una instruccion tipo 'call', 'ret', 'push', 'pop', 'enter', 'leave', etc. Si no tienes mucha idea, consulta cualquier buen libro de ensamblador (si eres muy novato, mejor evita los 'white papers' al principio y consiguete una buena introduccion).

Es tal la ola de ideologia RISC que nos invade, que posiblemente no necesites conocer algunas instrucciones 'vectoriales', tales como 'enter' y 'leave'. Algunos manuales de microprocesadores modernos desaconsejan su uso, pero como puedes ver (consulta el codigo en la seccion 1.1) el compilador 'gcc' sigue siendo adepto a ellas.

Ademas, en todos los siguientes codigos de ejemplo supondremos que se esta trabajando con ensamblador con preprocesador (extension '.S', consulta la seccion 1.4). Esto no es esencial, y es muy facil conseguir el codigo preprocesado (usa 'gcc -save-temps'), pero nos valdra como practica.

Las normas que vamos a exponer son validas para casi cualquier compilador de C, pero sobre estas cosas nunca puedes estar del todo seguro. Cuando menos, valen para gcc, y _deberian_ valer para los demas. Tambien son validas con pocas modificaciones para FORTRAN y Pascal.

2.1.1. Como llama el compilador de C a una subrutina.

Conceptualmente, las subrutinas son subprogramas a los que se llama desde un cierto programa, mandandoles una cierta cantidad de datos de manera que la subrutina devuelva uno o ningun dato.

Asi pues, la llamada a la subrutina tiene tres componentes:

a) El programa principal le envia ciertos datos.
b) El programa principal le cede el control.
c) La subrutina devuelve uno o ningun datos.

Estas componentes se gestionan como sigue:

-> Puesto que los datos que se envian a una subrutina en C son desechables, lo mas logico es introducir esos datos en la pila _antes_ de llamar a la subrutina. El propio modulo que llama a la subrutina se encarga luego de desechar los valores de la pila.

-> Puesto que la subrutina puede ser llamada desde varios modulos diferentes, la cesion de control (componente (b)) debe realizarse mediante una instruccion 'call'. La direccion de retorno quedara entonces contenida en la pila hasta que la subrutina devuelva el control al llamador, mediante un 'ret'.

-> Dado que el valor devuelto es tan solo de un tipo de dato (caracter, puntero, en coma flotante, etc.) es logico devolver ese dato en un registro, dado que la pila de la subrutina es desechada al salir de la misma. Tambien pueden devolverse datos de tipo mas complejo, como estructuras, pero no estudiaremos esto en el tutorial.

Por lo tanto, ya podemos hacer un esquema en ensamblador de como un cierto modulo llama a una subrutina. Los pasos a seguir deben ser:

1) Se meten en la pila los argumentos que se mandan a la subrutina. Por convenio, los argumentos se empujan de derecha a izquierda (es decir, en orden inverso a como es declarada la subrutina). Mas abajo tienes un ejemplo.

2) A continuacion se hace un 'call' a la subrutina. Esto cede el control a la misma.

3) Una vez que el modulo retoma el control, se eliminan

los argumentos empujados a la pila, usualmente con un
'subl CANTIDAD, %esp', donde CANTIDAD es el numero de bytes
reservados previamente (p. ej., CANTIDAD == $16, o lo que sea).
    4) Por convenio, el dato devuelto esta almacenado, segun
su tipo, en los siguientes registros:

Puntero	:	%eax
Caracter	:	%al
Palabra	:	%ax
Dpalabra	:	%eax
Qpalabra	:	%edx y %eax
Coma flotante	:	%st(0) (el TOS de la FPU)

    5) Opcionalmente, el compilador de C hace algunos chanchullos
para asegurarse de que la pila queda bien alineada en la subrutina.
No nos preocuparemos por esto (si tienes curiosidad por esto, mira
los comentarios que hicimos al respecto en 1.1).
    Con esto tenemos suficiente para dar un ejemplo: supongamos
que un programa en C desea llamar a la siguiente subrutina,

```
      extern struct barf * llama(long , char , short );
```

de manera que la subrutina devuelve un puntero, y tiene como
argumentos formales una palabra, un caracter y una dpalabra. En
nuestro codigo C podemos hacer la llamada con la instruccion:

```
        static long largo;
        static char corto;
        static short medio;
        static struct barf *ptr;
        main()
        {
            ...
            ptr = llama(largo, corto, medio);

            ...
        }
```

la llamada se compilaria como:

```
        ...

        pushw _medio
        pushb _corto
        pushl _largo         /* Hasta aqui 1) */
        call _llama          /* Esto es 2)    */
        addl $7, %esp        /* Esto es 3)    */
        movl %eax, _ptr      /* Esto es 4)    */
        ...
```

     Observa que hemos hecho que todas las variables enviadas
sean estaticas para facilitar la notacion. Si las variables
largo, medio, corto y _ptr fueran automaticas, las instrucciones
'push' estarian referidas al puntero de pila, que es donde se
almacenan las variables automaticas en un programa en C.
      Nota que la cantidad de bytes metidos en la pila es de
7 en total, que no es multiplo de 16. El compilador de C, para
mantener el alineamiento, posiblemente incluira una instruccion
'subl $9, %esp' antes del codigo que hemos descrito, y un
'addl $16, %esp' en vez del 'addl $7, %esp', todo ello con la
noble intencion de acelerar el codigo a costa de unos cuantos
bytes de pila.
    2.1.2. Como se comporta una subrutina en C.
    Ahora que ya sabemos lo que el compilador espera de nosotros,
es relativamente sencillo emular una subrutina C en ensamblador.
    En primer lugar, deberemos darle un nombre a la subrutina y
hacer que ese nombre sea visible para el linkador. Esto se logra
haciendo:

```
         .text
```

```
         .global _llama
         _llama:
```

al principio de la subrutina. Ahora el linkador ya sabe a donde tiene que saltar cuando se encuentre con el 'call _llama' en el modulo en C.

Una vez que el modulo nos ha cedido el control, tenemos que ocuparnos de lo que pasa en la pila, suponiendo que nuestra subrutina maneje algunos datos. Como suponemos que nuestra subrutina es la 'llama' definida mas arriba, sabemos que nos estan mandando 3 variables en la pila.

Justo en el momento en que el control se cede a nuestra subrutina, la estructura de la pila es la siguiente:

```
       ...          ...            (tamaño)
     9(%esp) --->  medio           (16 bits)
     8(%esp) --->  corto           ( 8 bits)
     4(%esp) --->  largo           (32 bits)
     0(%esp) --->  %eip de retorno (32 bits)
```

Para darse cuenta no hay mas que mirar como ingresaron los datos en la pila, en el codigo de ejemplo de la seccion 2.1.1.

Es con estos datos con los que tenemos que trabajar para desarrollar el codigo de la subrutina. Como veremos, hay dos estrategias: sacrificando un registro para acceder a la pila ordenadamente (modo 'canonico'), o accediendo a ella desordenadamente, pero ahorrando un registro (modo sin 'frame pointer').

UNA REGLA FUNDAMENTAL:

Como puedes ver, el modulo en C demuestra una extraordinaria confianza en las subrutinas a las que llama. El modulo llamador depende muy probablemente del contenido de registros como %ebp y %ebx (y quizas de muchos otros) para funcionar correctamente. Si al volver de la subrutina has modificado %ebp, %ebx u otro registro que use tu llamador, habras saboteado con toda seguridad su funcionamiento.

La norma acerca de registros para gcc es la siguiente:

```
NORMA: Los registros %esp, %ebp, %ebx, %esi, %edi _deben_
       ser devueltos intactos por toda subrutina. Lo mismo
       se aplica, por supuesto, a los registros de segmento.
COROLARIO: Puedes emplear libremente los registros %eax,
           %ecx, %edx en el interior de tu subrutina.
```

Olvidar esta regla es dejar una puerta abierta al caos, la desesperacion, la locura y lo peor de todo, el mensajillo 'segmentation fault - core dumped'.

UNA CONVENCION IMPORTANTE:

El compilador 'gcc' se emplea en muchos sistemas opertivos, como FreeBSD, NetBSD, linux, windows, etc. Lamentablemente, los que implementaron gcc en esos sistemas hicieron convenios distintos acerca de como se llama a las rutinas en ensamblador desde C.

El siguiente convenio es necesario para cygwin (implementacion de 'gcc' sobre windows), y posiblemente para otros sistemas:

Cuando programes una subrutina en ensamblador con la intencion de que se acceda a ella desde C, debes tener en cuenta que todos los simbolos que exportes con .globl o .global deben ir precedidos por un signo de subrayado '_'.

Por ejemplo, para definir una subrutina en ensamblador llamada 'procesa', que pueda ser llamada desde C, debes usar en tu programa ensamblador la directiva:

```
        .text
        .global _procesa
        _procesa:
         ...   (el resto de la subrutina)
```

Lo mismo vale para una variable estatica que quieras que sea visible desde C:

```
        .data
        .global _cadena
        _cadena:
        .asciz "Hola, mundo!\n"
```

Cualquier programa escrito en C podra acceder a 'procesa' o 'cadena' por su nombre, sin el subrayado, mientras que cualquier programa en ensamblador accedera a ellos con el nombre completo, con subrayado. Esto se ha pensado para evitar colisiones accidentales de nombres entre el compilador y el ensamblador.

En otras implementaciones de 'gcc' el subrayado puede no ser necesario. Para comprobarlo, basta que compiles con -S algun programa en C y veas como se declaran los nombres de las subrutinas.

Para curarnos en salud, en este tutorial hemos empleado el subrayado en todos los ejemplos.

---| 2.2 Subrutinas 'canonicas'.

Con la informacion obtenida en 2.1 ya tenemos suficentes datos como para empezar a programar subrutinas. Veamos el primer tipo.

Llamamos subrutinas 'canonicas' a las que se comportan de acuerdo con la tradicion de codificacion en C. Esta tradicion indica que hay que utilizar el registro %ebp para referirse a las variables locales, que estan en la pila. Se dice entonces que %ebp actua como 'frame pointer'. Si no se respeta este convenio, el debugger 'gdb' puede mostrarse algo confuso acerca del contenido de la pila de una subrutina.

La definicion de un nuevo 'frame pointer' se consigue con el siguiente mecanismo (siempre tomamos como ejemplo la subrutina 'llama' de la seccion anterior):

```
        .text

        .global _llama
        _llama:
        pushl %ebp       /* Almacenamos el viejo frame pointer. */
        movl %esp, %ebp  /* Nuevo frame pointer.                */
        subl $12, %esp   /* Espacio para variables automaticas. */
        ...              /* Resto de la subrutina.              */
        addl $12, %esp   /* Se liberan las variables automat.   */
        popl %ebp        /* Se recupera el viejo frame pointer. */
        ret              /* Retorno a modulo llamador.          */
```

las tres instrucciones finales pueden condensarse en dos:

```
        leave
        ret
```

esto es mas compacto, pero puede ser levemente ineficiente en algunas maquinas.

La idea que se persigue con el 'frame pointer' es tener un acceso facil a las variables automaticas de la subrutina. Si observamos la estructura de la pila despues de la instruccion 'subl $12, %esp', tenemos:

```
        ...            ...                  ...
     25(%esp) --->  (16 bits) medio  <---  13(%ebp)
     24(%esp) --->  ( 8 bits) corto  <---  12(%ebp)
     20(%esp) --->  (32 bits) largo  <---   8(%ebp)
     16(%esp) --->  %eip de retorno  <---   4(%ebp)
     12(%esp) --->  %ebp viejo       <---   0(%ebp)
      8(%esp) --->  (32 bits) var1   <---  -4(%ebp)
      4(%esp) --->  (32 bits) var2   <---  -8(%ebp)
      0(%esp) --->  (32 bits) var3   <--- -12(%ebp)
```

Hemos referido la pila a los dos punteros, %esp y %ebp. A las variables reservadas (tres enteros largos, 12 bytes en total) las hemos llamado var1, var2 y var3.

La ventaja de usar un 'frame pointer' para nuestra subrutina es evidente si suponemos que la pila se altera durante la

ejecucion de la subrutina. Por ejemplo, imaginemos que el
codigo de nuestra subrutina es:

```
        .text

        .global _llama
        _llama:
        pushl %ebp       /* Almacenamos el viejo frame pointer. */
        movl %esp, %ebp  /* Nuevo frame pointer.                */
        subl $12, %esp   /* Espacio para variables automaticas. */
        pushl %ebx       /* Conservamos para no corromperlo.    */
        xorl %ebx, %ebx  /* Resto de la subrutina ...           */
        ...
```

Despues de la instruccion 'pushl %ebx' el puntero de pila %esp ha disminuido en 4 unidades, por lo que ahora la variable 'var1' ha pasado a ser 12(%esp), ya no 8(%esp). Y el argumento 'corto' ha pasado a ser 28(%esp), cuando antes era 24(%esp).

Esto demuestra que encontrar las variables locales, o los argumentos de la subrutina, es bastante complicado si usas %esp, dado que tienes que tener una idea exacta del contenido de la pila _en_el_mismo_instante_de_acceder_a_ella_.

Sin embargo, dado que el puntero %ebp no ha cambiado durante el 'pushl %ebx', las variables y argumentos siguen teniendo la misma direccion que al principio. Es muy facil referirse a variables y argumentos a traves de %ebp.

NORMA: Si usas el 'frame pointer' pierdes un registro de proposito general pero ganas legibilidad y comodidad a la hora de acceder a variables automaticas y argumentos de la subrutina. Cualquier modificacion que hagas a un programa ensamblador _sin_ 'frame pointer' podria obligarte a revisar todo el codigo, si es que se produce una alteracion en el orden de la pila.

Debes calibrar cuidadosamente si quieres usar %ebp como 'frame pointer' o no. Las unicas ventajas son de legibilidad, mantenimiento del codigo, y debugging. No se obtienen ventajas evidentes en terminos de velocidad o espacio.

---| 2.3 Un ejemplo de subrutina 'canonica'.

Vamos a ver un ejemplo formado por un programa principal escrito en C y una subrutina en ensamblador. Lo que hara el programa sera pasar una cadena de caracteres a mayusculas, y devolver el numero de caracteres procesados en una variable tipo short.

Aqui tenemos el programa principal, main.c:

```
        /******** Aqui empieza el fichero main.c **********/
        #include<stdio.h>
        #include<stdlib.h>
        extern short procesa (char *);
        main()
        {
           char *c;
           short l;
           c = (char *) malloc(256 * sizeof(char));
           if (c == NULL) exit(1);
           printf("Introduce cadena (>255 caracteres): ");
           scanf("%255s", c);
           l = procesa(c);
           printf("\n\nCadena: %s\n", c);
           printf("Caracteres: %d\n", l);
           free(c);
           exit(0);
        }
        /*********** Fin de main.c *************/
```

y ahora el segundo fichero, sub.S:

```
         /********** Aqui empieza sub.S *********/
         #define ALINEA 16,,8
         .file "sub.S"
         .text
         .globl _procesa
         .align ALINEA
         _procesa:
         pushl %ebp
         movl %esp, %ebp
         subl $2, %esp             /* 2 bytes de variables,      */
                                   /* ver abajo.                 */
         pushl %ebx                /* Usaremos este registro.    */
         /* En este momento:
          *    -2(%ebp) == variable a devolver (16 bits).
          *     8(%ebp) == puntero a caracteres.
          */
          xorw %ax, %ax
          movw %ax, -2(%ebp)       /* Contador a cero.           */
          movl 8(%ebp), %eax       /* %eax apunta a la cadena.   */
          .align ALINEA            /* Alinea salto.              */
          L1:
          movb (%eax), %bl         /* Caracter en %bl.           */
          testb %bl, %bl           /* Fin de la cadena ?         */
          jz Lfin
          incw -2(%ebp)            /* Aumenta contador.          */
          andb $0xdf, (%eax)       /* Pasa a mayuscula.          */
          incl %eax                /* Siguiente elemento.        */
          jmp L1                   /* Cierra el lazo.            */
          .align ALINEA
          Lfin:
          movw -2(%ebp), %ax       /* Valor de retorno.          */
          popl %ebx                /* Recupera registro.         */
          addl $2, %esp            /* Libera vars. locales.      */
          popl %ebp                /* Libera stack frame.        */
          ret                      /* Hecho.                     */
         /************** Fin de sub.S  **************/
```

El programa es extraordinariamente tonto, pero consigue expresar la mayor parte de los conceptos que hemos visto. Observa en particular el uso de etiquetas locales, el uso de sufijos de todos los tipos en los operadores, y el uso del preprocesador. Tambien hemos usado un registro de mas, %ebx, que hemos tenido el cuidado de conservar.

Para compilar este programa, y ejecutarlo:

```
demeter# gcc -o cap main.c sub.S
demeter# ./cap
Introduce cadena (>256 caracteres):
HoLaYaDiOs

Cadena: HOLAYADIOS
Caracteres: 10
demeter#
```

Usando este ejemplo como punto de partida, puedes desarrollar casos mucho mas perfeccionados.

Ten en cuenta que el anterior ejemplo esta concebido como codigo de ejemplo, no como un codigo eficiente. Un par de correciones al mismo serian muy pertinentes. Por ejemplo, para mantener la pila alineada respecto a 4 bytes, se deberia usar ’subl $4, %esp’ en vez de ’subl $2, %esp’ al principio de la subrutina, incluso si de esos 4 bytes solo vamos a usar 2. Ademas, en vez de %ebx podriamos haber usado %edx, lo que nos habria ahorrado el andar conservando el registro en la pila. Mas aun, usando %ecx podriamos habernos

ahorrado el uso de la variable local. Pero con estas correciones creo que el codigo habria sido mucho menos instructivo.

---| 2.4 Subrutinas sin 'frame pointer'.

Entendiendo correctamente la seccion 2.2, es facil darse cuenta de en que consiste este tipo de subrutinas. Sencillamente, no empleamos el puntero %ebp como 'frame pointer', lo que exige usar algo mas el coco.

Como primer paso, reescribamos la subrutina del ejemplo anterior sin 'frame pointer'. Ademas de eliminar el uso de %ebp, cambiaremos de posicion el par de comandos 'pushl %ebx', 'popl %ebx' para que puedas apreciar como varia la manera de referirse a la variable local, que en unos momentos es (%esp) y en otros 4(%esp).

Para usar esta subrutina, guardala en un fichero ('sub2.S', por ejemplo), y compilala como la anterior.

```
          /*************** Aqui empieza sub2.S *************/

          #define ALINEA 16,,8
          .file "sub2.S"
          .text
          .globl _procesa
          .align ALINEA
          _procesa:

          subl $2, %esp            /* 2 bytes de variables,      */
                                   /* ver abajo.                 */
          /* En este momento, la estructura de la pila es:
           *
           *  6(%esp) --> puntero que nos han pasado.
           *  2(%esp) --> %eip de retorno.
           *  0(%esp) --> variable local (16 bits)
           */
          xorw %ax, %ax
          movw %ax, (%esp)         /* Contador a cero.           */
          movl 6(%esp), %eax       /* %eax apunta a la cadena.   */
          pushl %ebx               /* Usaremos este registro.    */
          /* La pila ha cambiado a:
           *
           * 10(%esp) -->  puntero que nos han pasado.
           *  6(%esp) -->  %eip de retorno
           *  4(%esp) -->  variable local (16 bits)
           *  0(%esp) -->  antiguo %ebx
           */
          .align ALINEA            /* Alinea salto.              */
          L1:
          movb (%eax), %bl         /* Caracter en %bl.           */
          testb %bl, %bl           /* Fin de la cadena ?         */
          jz Lfin
          incw 4(%esp)             /* Aumenta contador.          */
          andb $0xdf, (%eax)       /* Pasa a mayuscula.          */
          incl %eax                /* Siguiente elemento.        */
          jmp L1                   /* Cierra el lazo.            */
          .align ALINEA
          Lfin:
          popl %ebx                /* Recupera registro.         */
          /*  La pila ha vuelto a cambiar:
           *
           *  6(%esp) --> puntero que nos han pasado.
           *  2(%esp) --> %eip de retorno.
           *  0(%esp) --> variable local (16 bits)
           */
          movw (%esp), %ax         /* Valor de retorno.          */
```

```
          addl $2, %esp            /* Libera vars. locales.      */
          ret                      /* Hecho.                     */
         /************** Fin de sub2.S  **************/
```

Este programa apenas tiene un tamaño insignificantemente menor que el anterior, pero da una idea del metodo. Ahora todas las referencias de memoria son relativas a %esp, y dispondriamos del registro %ebp si estuvieramos realmente cortos de registros libres.

Cuantos mas 'push', 'pop', 'call' o cualquier otra instruccion que modifique %esp en medio del codigo, mas liante se vuelve la subrutina.

----------| PARTE 3 : Ensamblando 'inline' |----------

El ensamblado 'inline' consiste en la introduccion de codigo ensamblador en el interior de un codigo fuente en C. Esta tecnica es tan vieja como el lenguaje C, y el ensamblador 'gcc' posee unas capacidades extraordinarias para incluir ensamblador 'inline' en los programas en C, llegando a optimar la interaccion entre ambos lenguajes.

Todo esto viene a un precio. Hay que dar al compilador una informacion muy completa de los datos que nuestro codigo ensamblador usa como entrada, de los que usa como salida, y de los que usa como almacenamiento temporal. Sin esta informacion 'gcc' no podra optimar nuestro codigo y, lo que es peor, hasta puede ser inducido a error.

Cuando tiras una piedra en una laguna, no solo la zona de impacto es afectada por la piedra, si no que el efecto se expande por toda el agua. De la misma manera, introducir un codigo extraño en un programa C puede causar fallos donde menos lo esperas, si no avisas al compilador de lo que tu codigo hace exactamente.

En esta parte veremos dos tipos fundamentales de ensamblado dentro de C. El primero, al que llamaremos 'metodo Frankenstein', consiste en introducir el ensamblador a lo bruto, en plan protesis, usando algunas tecnicas elementales para asegurarte de que las cosas van a funcionar. El metodo Frankenstein es el que se usaba en los viejos tiempos. Su ventaja es que casa bien con la ley universal "cuando tengas dudas, usa la fuerza bruta"; es un metodo rapido y directo. Su desventaja es que nada te asegura que un cambio en la version del compilador, en las opciones de compilacion o en el codigo en C no vayan a desestabilizar completamente el programa.

El segundo metodo es el feten. Consiste en decirle a gcc en su propio lenguaje todo lo que necesita saber para que el trabajo quede bien hecho. Esta opcion tiene todas las ventajas, excepto que es un poco latosa de aprender. Bueno, tambien es menos divertida ;).

---| 3.1 El metodo Frankenstein.

La idea es simple, brutal y victoriana. Tomas los diferentes pedazos (brazos, piernas, placas de metal...), lo coses todo y lo echas a correr. El resultado suele ser feo y demoniaco, pero contundente. Como hemos aprendido en innumerables peliculas de serie B, eso de jugar a ser Dios conlleva el pago de un alto precio... asi que preparate para enfrentarte a los monstruos que vas a crear.

Bromas aparte, el metodo Frankenstein consiste en los siguientes pasos:

1) Escribe el codigo en C que quieres modificar.

2) Compilalo con 'gcc -S' y localiza la zona en la que quieres injertar el ensamblador.

3) Observa bien la zona de implantacion para asegurarte de que el injerto sera compatible con el codigo.

4) Injerta el ensamblador en el codigo C. Esto se hara usualmente en forma monolitica (tus instrucciones en ensamblador no deberan ser procesadas por el compilador, que se limitara a pasarlas al ensamblador 'tal cual').

5) Compila el codigo injertado con ’gcc -S’ y comprueba que la cosa marcha como tu quieres.
6) Ahora compila y linka el programa injertado.
7) Ejecutalo. Si funciona, debes lanzar el grito ritual: "Estaaaaa viiiiivooo! Muhahahahah!" Ya estas listo para castigar al mundo con tu creacion infernal.

3.1.1 Insercion de ensamblador.

Para insertar ensamblador en un codigo en C hay que utilizar el comando ’asm’. Sin embargo, para evitar advertencias por parte del compilador (por ejemplo, compilando con comaptibilidad ANSI), es mejor usar ’__asm__’, con dos subrayados antes y despues de la directiva.

Como primer ejemplo, imaginemos que queremos introducir una etiqueta inocua en el codigo en C, de manera que al ejecutar ’gdb’ podamos saltar rapidamente a esa parte de codigo. Si tenemos un programa como el siguiente (guardalo en un fichero ’memo.c’):

```
#include <stdio.h>
main (int argc, char *argv[])
{
    int i;
    puts("Este programa ha sido invocado como:");
    for (i = 0 ; i < argc ; i++)
    {
        printf("%s", argv[i]);
        putchar(’ ’);
    }
    putchar(’\n’);
    exit(0);
}
```

otro de nuestros programas tontos patentados. Lo que hace esta claro:

```
demeter# gcc -o memo memo.c

demeter# ./memo jarl de peich morl!
Este programa ha sido invocado como:
/root/tut/memo jarl de peich morl!
demeter# ./memo *.c
Este programa ha sido invocado como:
/root/tut/memo bitm.c main.c memo.c test.c
demeter#
```

Etcetera. Ahora supongamos que queremos definir un ’breakpoint’ para gdb entre el ’printf’ y el ’putchar’ del bucle. Para ello introducimos entre ambas instrucciones un fragmento de ensamblador:

```
...
printf("%s", argv[i]);
__asm__(" .global _brk ; _brk: ");
putchar(’ ’);
...
```

Bien, ahora volvemos a compilar y ejecutamos gdb:

```
demeter# gcc -o memo memo.c
demeter# gdb memo
(gdb saluda y se identifica)
(gdb) disassemble brk
Dump of assembler code for function brk:
<brk>:    add $0xfffffff4, %esp
<brk+3>:  call <putchar>
... (mas lineas en ensamblador)
End of assembler dump.
(gdb)
```

Este diminuto (e inocuo) parche en ensamblador nos permite fijar la atencion en la parte del fichero que nos interesa. Como primer ejemplo de inoculacion de codigo, es bastante util.

```
    En general, para insertar lineas mas largas de ensamblador
todo lo que necesitas es usar habilmente el separador de comandos
';' de gas y el simbolo de union de lineas '\' del compilador.
Un ejemplo hipotetico seria:
       ... (codigo en C)
       __asm__(" .globl _franky       ;" \
               " _franky:             ;" \
               "      addl $4, %esp   ;" \
               "      incl %eax       ;" \
               "      leave           ;" \
               "      ret              ");

       ... (codigo en C)
Las normas son:
      1) Cada comando en ensamblador va entre comillas.
      2) Usa un ';' para separar los comandos.
Si compilas con 'gcc -S' veras que este codigo injertado se
escribe en una sola linea, con los ';' como separadores. Si
quieres que al hacer 'gcc -S' el codigo en ensamblador se
vea mas agradable, en lineas separadas, en vez del ';' usa
el par '\n\t'. Por ejemplo:
       __asm__(" .globl _franky       \n\t" \
               " _franky:             \n\t" \
               "      addl $4, %esp   \n\t" \
               "      incl %eax       \n\t" \
               "      leave           \n\t" \
               "      ret              ");
Aunque es algo molesto de escribir, el resultado estetico
vale sobradamente la pena.
    Si has compilado con '-S' alguno de los ejemplos anteriores,
habras notado que el codigo injertado aparece en el ensamblador
entre los comandos '/APP' y '/NO_APP', por ejemplo:
       /APP
           .globl _brk ; _brk:
       /NO_APP
Estos comandos le dicen al ensamblador que las lineas que has
injertado han sido escritas por un humano, por lo que el
ensamblador debe tener especial cuidado al leerlas. A donde
hemos ido a parar. Los ordenadores nos tratan con condescendencia!
En todo caso, gracias a esto podemos localizar rapidamente los
comandos injertados en el fichero '.s'.
    Resumiendo:
      1) Para injertar ensamblador en gcc, basta con usar
la directiva  __asm__(" ... "); .
      2) La cadena entre parentesis se puede romper en varias,
que se consideraran parte de una sola linea del ensamblador. En
forma general:  __asm__(" ... " " ... "  ...  " ... "); .
      3) Para separar comandos de ensablador, o bien usas ';',
lo cual es sencillo pero queda feo en el fichero '.s', o bien
usas '\n\t', que te hace escribir algo mas pero resulta mas
bonito en el '.s'.
      4) Tu codigo injertado va en el fichero '.s' entre los
comandos '\APP' y '\NO_APP'.
    3.1.2  Injertos monoliticos.
    El compilador 'gcc' es lo bastante inteligente como para darse
cuenta de que ciertas lineas en ensamblador que introduces no son
necesarias, lo cual le puede llevar a eliminarlas. Tambien puede
ocurrir que decida mezclar tus lineas en ensamblador con las que
el genera, para acelerar el codigo.
    Si estas haciendo un injerto a lo bruto, esto puede ser de lo
mas indeseable. Para decirle al compilador que nuestro codigo es
peligroso se emplea el comando '__volatile__', justo despues del
```

comando '__asm__'. Es decir:

```
      __asm__ __volatile__(" ... "),
```

donde la (o las) cadenas en el interior de los parentesis van exactamente igual que las descritas en la seccion anterior.

El compilador entiende el '__volatile__' como un signo de peligro y no tocara lo que haya dentro, en la medida de lo posible.

3.1.3 Un ejemplo retorcido.

Naturalmente. Los ejemplos de injertos son siempre retorcidos.

Para apartarnos un poco de nuestra politica de programas idiotas, consideremos un ejemplo un poco mas pintoresco. El siguiente programa genera un fichero llamado 'sal.bmp', que contiene el dibujo de una espiral de Arquimedes, en blanco y negro y resolucion de 300x300 pixels.

Almacena el siguiente programa en un fichero ('bitm.c'):

```
/************ Aqui comienza 'bitm.c' ***************/
#include <stdio.h>
#include <math.h>
/* Cabecera de un fichero BMP 300x300 monocromo. */
static char head[] = { \
0x42,0x4d,0x1e,0x2f,0,0,0,0,0,0,0x3e,0,0,0,0x28,0, \
0,0,0x2c,0x01,0,0,0x2c,0x01,0,0,0x01,0,0x01,0,0,0, \
0,0,0xe0,0x2e,0,0,0xc4,0x0e,0,0,0xc4,0x0e,0,0,0,0, \
0,0,0,0,0,0,0,0,0,0,0xff,0xff,0xff,0};
/* Mascara que localiza un bit en un byte. */
static char masc[] = {128,64,32,16,8,4,2,1};
/* Constantes del BMP. */
#define TAM (40*300)
#define LIN 40
#define HDSIZ 62
#define RADIO 3.0   /* Escala del radio de la espiral. */
main()
{
     char *imag, *alias;
     FILE *fich;
     unsigned long i, j;
     double x,y,t;
     if ((fich = fopen("sal.bmp","w")) == 0)
         { puts("Error: fopen"); exit(1); }
     imag = (char *) malloc(TAM * sizeof(char));
     if (imag == NULL) { puts("Error: malloc"); exit(1); }
     /* Escribe cabecera. */
     for (i = 0 ; i < HDSIZ ; i++)
         putc(head[i], fich);
     /* Inicializa imagen */

     alias = imag;
     for (i = 0 ; i < 300 ; i++)
     {
         for (j = 0 ; j < (LIN-3) ; j++)
             *(alias++) = 0xff;
         *(alias++) = 0xf0;
         *(alias++) = 0;
         *(alias++) = 0;
     }
     /* Genera imagen.                    */
     for (t = 0 ; t < 36 ; t += 0.001)
     {
     /* Calculamos coords. de la espiral. */
        x = 150.0 + RADIO * t * cos(t);
        y = 150.0 + RADIO * t * sin(t);
     /* Eliminamos puntos fuera del BMP.  */
```

```
            if ((x > 300) || (x < 0)) continue;
            if ((y > 300) || (y < 0)) continue;
         /* Incluimos el punto en la imagen.  */
            j = (unsigned long) y;
            i = (unsigned long) (x/8);
            imag[(LIN*j)+i] &= ~masc[((unsigned long) x)%8];
         }
         /* Escribe imagen en fichero.        */
         for (i = 0 ; i < TAM ; i++)
             putc(imag[i], fich);
         free(imag);
         fclose(fich);
         exit(0);
      }
      /***************** Fin de 'bitm.c ****************/
```

Supongamos que queremos sustituir el calculo de las coordenadas de la espiral por una version en ensamblador. Nuestra primera tarea es localizar la zona de injerto, lo cual se puede lograr sencillamente intercalando un par de etiquetas que pasaran al ensamblador:

```
      ...
      __asm__ __volatile__("comienzo:");
      x = 150.0 + RADIO * t * cos(t);
      y = 150.0 + RADIO * t * sin(t);
      __asm__ __volatile__("final:");
      ...
```

Ahora compilamos con 'gcc -ffast-math -S' y hacemos una busqueda de la zona entre las etiquetas. El uso de la opcion '-ffast-math' ayuda a simplificar el codigo que queremos localizar. Veamos que es lo que encontramos:

```
       /APP
             comienzo:
       /NO_APP
             addl $-8,%esp
             fldl -48(%ebp)
             subl $8,%esp
             fstpl (%esp)
             call _cos
             addl $16,%esp
             fldl LC5
             fmulp %st,%st(1)
             fmull -48(%ebp)
             fldl LC6
             faddp %st,%st(1)
             fstpl -32(%ebp)
             addl $-8,%esp
             fldl -48(%ebp)
             subl $8,%esp
             fstpl (%esp)
             call _sin
             addl $16,%esp
             fldl LC5
             fmulp %st,%st(1)
             fmull -48(%ebp)
             fldl LC6
             faddp %st,%st(1)
             fstpl -40(%ebp)
       /APP
               final:
       /NO_APP
```

Este codigo nos permite formular algunas conjeturas:

1) -32(%ebp) almacena la variable x.

```
          2) -40(%ebp) almacena la variable y.
          3) -48(%ebp) almacena la variable t.
          4) LC5 almacena la constante RADIO (que vale 3.0)
          5) LC6 almacena la constante 150.0
```
que es mas o menos todo lo que necesitamos para desarrollar nuestro codigo. Pero antes de comenzar, observemos algo que puede causarnos muchos problemas:
```
-> Las constantes LC5 y LC6 han sido definidas por el
-> compilador precisamente porque las hemos usado en las
-> lineas de codigo que queremos sustituir. Asi que si ahora
-> eliminamos esas lineas, LC5 y LC6 ya no existiran, o bien
-> estaran asociados a otras constantes. En consecuencia,
-> nuestro codigo ensamblador debe definir estas dos
-> constantes (con otro nombre que no interfiera con C).
```
En esencia, tendremos que definir dos cantidades de doble precision en nuestro codigo ensamblador. Una sera 3.0 y la otra 150.0. Este tipo de cosas son las que te hacen amar los chanchullos con ensamblador.

Ahora que ya nos hemos salvado de este diabolico efecto colateral, pasemos a construir un codigo que haga el calculo de las cantidades que nos interesan. Por ejemplo, esto podria servir:

```
    .data
    Lradio:
        .double 3.0
    Lcentro:
        .double 150.0
    .text
                          /*       Pila de la FPU          */
        fldl -48(%ebp)    /* t                             */
        fld %st(0)        /* t : t                         */
        fldl Lradio       /* 3 : t : t                     */
        fmulp %st(2)      /* t : 3t                        */
        fsincos           /* Cos(t) : Sen(t) : 3t          */
        fmul %st(2)       /* 3tCos(t) : Sen(t) :3t         */
        fxch %st(2)       /* 3t : Sen(t) : 3tCos(t)        */
        fmulp %st(1)      /* 3tSen(t) : 3tCos(t)           */
        fldl Lcentro      /* 150 : 3tSen(t) : 3tCos(t)     */
        fadd %st(0),%st(2) /* 150 : 3tSen(t) : 150+3tCos(t) */
        faddp %st(1)      /* 150+3tSen(t) : 150+3tCos(t)   */
        fstpl -40(%ebp)   /* 150+3tCos(t)                  */
        fstpl -32(%ebp)
```
Lo cual, insertado en modo ensamblador seria:
```
    __asm__ __volatile__(" .data                    \n\t" \
                         " Lradio:                  \n\t" \
                         "     .double 3.0          \n\t" \
                         " Lcentro:                 \n\t" \
                         "     .double 150.0        \n\t" \
                         " .text                    \n\t" \
                         "     fldl  -48(%ebp)      \n\t" \
                         "     fld  %st(0)          \n\t" \
                         "     fldl  Lradio         \n\t" \
                         "     fmulp  %st(2)        \n\t" \
                         "     fsincos              \n\t" \
                         "     fmul  %st(2)         \n\t" \
                         "     fxch  %st(2)         \n\t" \
                         "     fmulp  %st(1)        \n\t" \
                         "     fldl  Lcentro        \n\t" \
                         "     fadd  %st(0), %st(2) \n\t" \
                         "     faddp  %st(1)        \n\t" \
                         "     fstpl  -40(%ebp)     \n\t" \
```

```
                        "     fstpl  -32(%ebp)     \n\t");
```

Prueba a sustituir las dos lineas en C que definen 'x' e 'y' por este injerto en ensamblador, y compila el resultado. Ten en cuenta que tu compilador podria almacenar 'x', 'y' y 't' en otras posiciones. Salvo esto, el codigo deberia funcionar de manera general.

Por si te has perdido, he aqui la forma final del codigo mixto para el programa en C anterior:

```
/***************** Aqui comienza bitm2.c ********************/
#include <stdio.h>
#include <math.h>
/* Cabecera de un fichero BMP 300x300 monocromo. */
static char head[] = { \
0x42,0x4d,0x1e,0x2f,0,0,0,0,0,0x3e,0,0,0,0x28,0, \
0,0,0x2c,0x01,0,0,0x2c,0x01,0,0,0x01,0,0x01,0,0,0, \
0,0,0xe0,0x2e,0,0,0xc4,0x0e,0,0,0xc4,0x0e,0,0,0,0, \
0,0,0,0,0,0,0,0,0,0,0xff,0xff,0xff,0};
/* Mascara que localiza un bit en un byte. */
static char masc[] = {128,64,32,16,8,4,2,1};
/* Constantes del BMP. */
#define TAM (40*300)
#define LIN 40
#define HDSIZ 62
#define RADIO 3.0   /* Escala del radio de la espiral. */
main()
{
    char *imag, *alias;
    FILE *fich;
    unsigned long i, j;
    double x,y,t;
    if ((fich = fopen("sal.bmp","w")) == 0)
        { puts("Error: fopen"); exit(1); }
    imag = (char *) malloc(TAM * sizeof(char));
    if (imag == NULL) { puts("Error: malloc"); exit(1); }
    /* Escribe cabecera. */
    for (i = 0 ; i < HDSIZ ; i++)
        putc(head[i], fich);
    /* Inicializa imagen */
    alias = imag;
    for (i = 0 ; i < 300 ; i++)
    {
        for (j = 0 ; j < (LIN-3) ; j++)
            *(alias++) = 0xff;
        *(alias++) = 0xf0;
        *(alias++) = 0;
        *(alias++) = 0;
    }
    /* Genera imagen.                    */
    for (t = 0 ; t < 36 ; t += 0.001)
    {
    /* Calculamos coords. de la espiral. */
    __asm__ __volatile__(" .data                    \n\t" \
                         " Lradio:                  \n\t" \
                         "     .double 3.0          \n\t" \
                         " Lcentro:                 \n\t" \
                         "     .double 150.0        \n\t" \
                         " .text                    \n\t" \
                         "     fldl  -48(%ebp)      \n\t" \
                         "     fld  %st(0)          \n\t" \
                         "     fldl  Lradio         \n\t" \
                         "     fmulp  %st(2)        \n\t" \
                         "     fsincos              \n\t" \
```

```
                        "     fmul  %st(2)         \n\t" \
                        "     fxch  %st(2)         \n\t" \
                        "     fmulp  %st(1)        \n\t" \
                        "     fldl  Lcentro        \n\t" \
                        "     fadd  %st(0), %st(2) \n\t" \
                        "     faddp  %st(1)        \n\t" \
                        "     fstpl  -40(%ebp)     \n\t" \
                        "     fstpl  -32(%ebp)     \n\t");
    /* Eliminamos puntos fuera del BMP.  */
       if ((x > 300) || (x < 0)) continue;
       if ((y > 300) || (y < 0)) continue;
    /* Incluimos el punto en la imagen.  */
       j = (unsigned long) y;
       i = (unsigned long) (x/8);
       imag[(LIN*j)+i] &= ~masc[((unsigned long) x)%8];
    }
    /* Escribe imagen en fichero.        */
    for (i = 0 ; i < TAM ; i++)
        putc(imag[i], fich);
    free(imag);
    fclose(fich);
    exit(0);
}
/********************* Fin de bitm2.c *************************/
```

---| 3.2 Introduccion al metodo ortodoxo.

El metodo Frankenstein esta sujeto a muchos fallos y es muy dificil de mantener. Si recordamos el ejemplo de la seccion 3.1.3, los problemas que afrontamos fueron:

1) Localizar las variables de entrada para nuestro codigo.
2) Localizar las variables de salida para nuestro codigo.
3) Asegurarnos de que no modificamos accidentalmente ningun dato necesario para el compilador.

El metodo ’ortodoxo’ de ensamblado inline nos permite instruir a ’gcc’ para que resuelva automaticamente estos tres problemas.

Intuitivamente, puedes suponer que se produce la siguiente situacion: tu codigo, con su informacion sobre 1), 2) y 3), es como una burbuja de codigo ensamblador metida entre el codigo ensamblador generado por ’gcc’. Dado que se tiene toda la informacion sobre como interactua ese codigo, en una fase avanzada de la compilacion el gcc rompe la burbuja de tu codigo y lo mezcla todo, optimando el resultado total.

La desventaja del metodo Frankestein es que la burbuja injertada es irrompible y, si entra en conflicto con el compilador, puede llegar a convertirse en un cancer para el codigo.

La anterior metafora puede hacernos suponer que nos vamos a comunicar con el compilador a un nivel bastante bajo. Tendremos que darle los datos bastante masticados para que los entienda. La cosa puede ser complicada si queremos injertar fragmentos de codigo muy largos; pero recuerda que estos ultimos casos los puedes meter en subrutinas (ver Parte 2).

El tema es delicado, asi que comencemos con algunos casos simples antes de pasar a la situacion general. Lee esta seccion de una vez, intentando captar la idea general; mas adelante se iran perfilando los detalles.

En primer lugar, recordemos el codigo que usamos en la seccion 3.1 para incluir etiquetas en un programa C:

```
       __asm__(" .globl _brk ; _brk: ");
```

Este injerto no tiene datos de entrada, no tiene datos de salida y no modifica el contenido de ninguna variable ni registro: sencillamente es una directiva y una etiqueta. Toda la informacion generada son algunos simbolos para el linkador,

asi que no nos tenemos por que preocupar de que el compilador se indigeste. Este codigo no necesita modificacion para ser 'ortodoxo'.

Vamos a ver un ejemplo mucho mas ambicioso. Recuerda el ejemplo de 3.1.3, el codigo en coma flotante. Alli nuestro problema era que el codigo necesitaba incorporar ciertas variables de C, y no sabiamos muy bien como hacerlo. Observa que diferente es la aproximacion ortodoxa de la Frankestein: Sustituyamos el calculo de las coordenadas de la espiral en 'bitm.c' por las tres lineas (guardalo en 'bitm3.c'):

```
__asm__("fsincos" : "=t" (x), "=u" (y) : "0" (t));
x =  (x * t * RADIO) + 150.0;
y =  (y * t * RADIO) + 150.0;
```

Lo que hemos hecho es:

a) Ejecutamos 'fsincos', con dato de entrada 't' y datos de salida 'x' e 'y'. Indicar al compilador estas entradas y salidas se logra con los comandos que van con los ':' al final de la primera linea (veremos la sintaxis de estos comandos algo mas abajo; por ahora olvidate de ello).

b) Ahora que x == Sen(t) e y == Cos(t), hacemos el calculo normal de las coordenadas, en las lineas restantes. La ventaja de las tres lineas que tenemos arriba es que podemos compilar el codigo con cualquier nivel de optimacion, dado que el compilador se encargara de hacer los arreglos. Por ejemplo, usando 'gcc -O6' el codigo marcha a la perfeccion. Dado que a este nivel de optimacion el compilador opera perfectamente con sumas y productos, la unica optimacion relevante es la que le hemos pedido, es decir, que use 'fsincos'. Compila este programa con 'gcc -O6 -S' y observa hasta que punto optima el compilador el resto del codigo. Se ha logrado un resultado al menos tan bueno como nuestro Frankestein sin necesidad de escribir mas que una linea en ensamblador.

La idea del ensamblador inline bien hecho se reduce muchas veces a insinuar a 'gcc' que debe usar un comando complicado (como 'fsincos'), dejandole que haga el resto del trabajo pesado. Tambien puedes introducir fragmentos largos de ensamblador en el codigo, pero puede resultar una mala politica. Lograr una eficiencia como la de gcc con instrucciones de proposito general puede llevarte un tiempo que puedes emplear en otras cosas mejores.

Veamos otro ejemplo, que personalmente siempre he echado de menos en los lenguajes de alto nivel. A menudo, cuando operamos con C queremos saber si cierta instruccion de suma ha producido un acarreo, por ejemplo cuando se opera con enteros en precision arbitraria. Veamos una solucion al problema del acarreo:

```
#include <stdio.h>
main()
{
    unsigned short i, j, sum, acarreo;
    i = 40000;
    j = 50000;
    printf("Sumando %d y %d:\n",i,j);
    __asm__(" xorw %0, %0     \n\t" \
            " xorw %1, %1     \n\t" \
            " movw %2, %1     \n\t" \
            " addw %3, %1     \n\t" \
            " adcw %0, %0     "
            : "=r" (acarreo) , "=r" (sum)
            : "rm" (i) , "rm" (j) );
    printf("\nSuma   : %d\n", sum);
    printf("Acarreo: %d\n", acarreo);
    exit(0);
```

```
        }
```

Introduce este codigo en un fichero ('aca.c') y compilalo:

```
        demeter# gcc -o aca aca.c
        demeter# ./aca
        Sumando 40000 y 50000
        Suma   : 24464
        Acarreo: 1
        demeter#
```

Una vez mas es un ejemplo tonto, pero que implementa una funcionalidad que a C le resulta dificil imitar. Observa que los comandos en ensamblador operan sobre %0, %1, %2 y %3. Esto es natural, dado que no sabemos como va a llamar el compilador a los argumentos de las instrucciones en ensamblador: la solucion es darles los nombres simbolicos %0, %1, %2 y %3. La informacion que se incluye al final se encargara de decirle al ensamblador como sustituir esos simbolos.

Veamos brevemente como compila esto gcc. La zona interesante es la siguiente (haz un 'gcc -S'):

```
        /APP
             xorw %dx, %dx
             xorw %ax, %ax
             movw -2(%ebp), %ax
             addw -4(%ebp), %ax
             adcw %dx, %dx
        /NO_APP
```

De esto podemos deducir que el compilador ha hecho las sustituciones:

```
             %0  ==  %dx
             %1  ==  %ax
             %2  ==  -2(%ebp)  ( la variable 'i' )
             %3  ==  -4(%ebp)  ( la variable 'j' )
```

Siguiendo el codigo en ensamblador un poco mas, podemos deducir que %ax va a parar a la variable 'sum' y que %dx va a parar a 'acarreo'.

?Como se corresponde esto con las sugerencias que hemos hecho en el __asm__? Veamos la sintaxis y el significado de las mismas. Tenemos:

```
             : "=r" (acarreo) , "=r" (sum)
             : "rm" (i) , "rm" (j)
```

a) La primera linea corresponde a los argumentos de salida del injerto, en tanto que la segunda linea corresponde a las entradas. Cada linea va precedida siempre por un ':'. Si no hubiera salidas, aun asi habria que poner el ':', para que el compilador no se confunda.

b) Los argumentos dentro de cada linea (sean entradas o salidas) son separados por comas, en caso de haber varios.

c) Cada argumento consta de dos partes:

c1) La primera, entre comillas, indica como se escribe el argumento dentro del injerto en ensamblador; es decir, cuando se sustituyen %0, %1, etc., el compilador debe saber si cada uno de esos simbolos tiene que sustituirse por un registro, por una referencia a memoria, por un registro de la FPU, por una constante, etc.

c2) La segunda, entre parentesis, indica a que variable del programa en C se corresponde esa entrada o salida.

Ejemplos:

```
        "rm" (i)   ---  esto indica que la variable 'i' debe
                        meterse en el injerto como una referencia
                        a memoria o como un registro, que es lo
                        significa la cadena "rm".
```

```
        "=r" (sum) ---  esto indica que la variable 'sum' debe
                        meterse en el injerto como un registro.
                        Por convenio, toda salida lleva siempre
                        un signo '=' metido en la primera parte.
                        Asi, 'registro y salida' se escribe "=r".
          Mas adelante veremos las posibilidades mas comunes para
          la parte c1).
    d) El compilador asocia los simbolos %0, %1, %2, %3... a los
argumentos de entrada y salida por orden de aparicion. Asi, tenemos
que:
  Simbolo: --> Corresponde a: -> En el injerto:  -->   En el .s:
    %0         "=r" (acarreo)    sera un registro      %dx
    %1         "=r" (sum)        sera un registro      %ax
    %2         "rm" (i)          registro o memoria    -2(%ebp)
    %3         "rm" (j)          registro o memoria    -4(%ebp)
dado que es en este orden en el que hemos introducido los datos
en las lineas de entrada y salida.
    Como puedes ver, el compilador tiene a veces opciones para
elegir, como en el caso "rm": es gcc quien decide si el simbolo
%2 debe ser rellenado con un registro o con una referencia a
memoria, segun convenga. En nuestro ejemplo se ha inclinado por
una referencia a memoria. Es aconsejable dejar esta flexibilidad
al compilador, puesto que los registros no son muy abundantes
en los ix86.
    A veces es necesario restringir estas opciones. Por ejemplo,
en las entradas solo permitimos referencias a registros, dado
que la instruccion 'xor %0, %0', por ejemplo, no podria aceptar
que se sustituyese %0 por una referencia a memoria. Este tipo
de instrucciones con argumento repetido solo funcionan con
registros (por ejemplo, 'xor (%ebp), (%ebp)' es ilegal para los
microprocesadores ix86).
    De acuerdo. La primera vez que uno lee esto resulta muy
complicado. Mi sugerencia es que recuerdes que en realidad le
estamos comunicando al compilador una informacion muy sencilla:
entradas y salidas. Ponte en el lugar del compilador e imagina
que es lo que necesitas saber para cada entrada o salida:
       1) A que variable de C corresponde esa entrada o salida.
       2) Que simbolo asocio a esa entrada o salida.
       3) Como sustituyo ese simbolo en el codigo ensamblador.
Nada mas. Ahora hay que hacerse a la curiosa notacion que han
escogido los desarrolladores de gcc para especificar esta
informacion. Pero esto no es mas que la parte burocratica;
echale un poco de tiempo y basta.
    Antes de entrar en una descripcion mas detallada del formato,
veamos un ultimo ejemplo. En la primera parte de este tutorial
hablamos de la manera de introducir instrucciones no soportadas
por el ensamblador, tales como 'rdtsc'. Esta instruccion, presente
en la mayor parte de los micros de la serie i686, carga en el par
de registros %edx:%eax la cantidad de ciclos de reloj transcurridos
desde el arranque del microprocesador (modulo 2 elevado a 64). Esto
quiere decir que, por ejemplo, el contador de rdtsc (llamado TSC,
'time stamp counter') se incrementa en 700 millones de unidades cada
segundo en mi K7-700. La instruccion rdtsc permite hacer mediciones de
tiempo muy precisas, o calcular el numero de ciclos de reloj transcu-
rridos, aproximadamente, entre dos eventos.
    Para implementar 'rdtsc' en ensamblador inline, tengamos en
cuenta que esta instruccion no tiene argumentos de entrada, no
corrompe ningun registro, y como registros de salida tiene
especificamente a %edx y %eax. Si miramos la documentacion de
'gcc' (www.gnu.org), en la seccion de 'Extensiones al lenguaje C',
subseccion de 'Restricciones para maquinas concretas' (en ingles),
podemos encontrar que la manera de indicar al gcc que una entrada
```

o salida corresponde al par %edx:%eax es usando "A". Por tanto, el comando ensamblador sera:

```
                  /* rdtsc */       /* salidas */
      __asm__(".byte 0x0f, 0x31" : "=A" (lectura));
```

Donde 'lectura' sera la variable que nos interese. Veamos un ejemplo:

```
        #include <stdio.h>
        #include <math.h>
        main()
        {
            unsigned long long lectura1, lectura2;
            double x,y;
            __asm__(".byte 0x0f, 0x31" : "=A" (lectura1));

            for (x = 0.0 ; x < 10.0 ; x += 0.01)
                y = exp(x);
            __asm__(".byte 0x0f, 0x31" : "=A" (lectura2));

            lectura2 = (lectdura2 - lectura1) / 1000;
            printf("Numero de ciclos: %d\n", lectura2);
            exit(0);
         }
```

Tras guardar esto en el fichero 'rdtsc.c', lo compilamos y ejecutamos:

```
        demeter# gcc -lm -o rdtsc rdtsc.c
        demeter# ./rdtsc
        Numero de ciclos: 230
        demeter#
```

Es decir, que al micro le lleva en torno a 230 ciclos el ejecutar cada exponencial en el bucle, lo cual es razonable teniendo en cuenta que cada llamada a 'exp' es canalizada por la libreria matematica.

NOTA : Como curiosidad, he compilado y ejecutado este programa en linux, FreeBSD y Windows 98. En los dos primeros los tiempos de ejecucion varian entre 210 y 240 ciclos, en tanto que en Win98 el tiempo ronda los 310 ciclos. La libreria matematica de Win98 no es todo lo eficiente que se podria desear.

Veamos que sucede si calculamos la exponencial directamente en ensamblador. Si sustituimos la linea 'y = exp(x)' por un equivalente en ensamblador, obtenemos:

```
        #include <stdio.h>
        main()
        {
             unsigned long long lectura1, lectura2;
             double x,y;
             __asm__(".byte 0x0f, 0x31" : "=A" (lectura1));
             for (x = 0.0 ; x < 10.0 ; x += 0.01)
             {
                __asm__(" fldl2e        \n\t" \
                        " fmulp         \n\t" \
                        " fld %0        \n\t" \
                        " fld %0        \n\t" \
                        " frndint       \n\t" \
                        " fsubrp        \n\t" \
                        " f2xm1         \n\t" \
                        " fld1          \n\t" \
                        " faddp         \n\t" \
                        " fscale        \n\t" \
                        " fxch          \n\t" \
                        " fstp %0       \n\t"
                        : "=t" (y)
```

```
                        : "0" (x) );
            }
            __asm__(".byte 0x0f, 0x31" : "=A" (lectura2));
            lectura2 = (lectura2 - lectura1) / 1000;
            printf("Numero de ciclos: %d\n", lectura2);
            exit(0);
       }
```

Guardando el resultado en 'rdtsc2.c' y compilando, obtenemos:

```
       demeter# ./rdtsc2
       Numero de ciclos: 119
```

Lo que ciertamente es una mejora. Como es natural, la libreria matematica da ciertas seguridades (comprobacion de errores sobre todo) que no se deben despreciar a la ligera. Pero si tus calculos no son de caracter cientifico, siempre te puedes apoyar en el ensamblador para acelerar enormemente tus programas de calculo en coma flotante.

Si observas el ejemplo de arriba, lo que tenemos es en la practica un hibrido entre ensamblador y C. El C se encarga de gestionar variables y estructuras reiterativas mientras nosotros implementamos lo interesante en ensamblador. Vaaale, quizas este tipo de programas no son muy portables, pero a fin de cuentas, como dice el Fortune File, "Portable == Inutil en cualquier maquina".

---| 3.3 Implementacion del metodo ortodoxo.

Ocupemonos pues de la implementacion del ensamblado 'inline' como Dios manda. El formato general del comando __asm__ es el siguiente:

```
   __asm__ (" ... " : [entradas] : [salidas] : [corrupto] );
```

Donde:

```
      [entradas] : Se refiere a las entradas en el codigo
                   ensamblador.
      [salidas]  : Se refiere a las salidas del codigo
                   ensamblador.
      [corrupto] : Se refiere a los datos/registros corrompidos
                   por el codigo ensamblador. En 'jargon', corrupto
                   se traduce como 'clobbered' (vapuleado).
```

Cada uno de estos tres campos puede constar de cero o mas partes, cada una de ellas denotando una correspondencia entre un simbolo en el codigo ensamblado y un dato en el programa en C. En caso de haber varias partes en un campo, se separan mediante comas.

Cada parte en los dos primeros campos (entradas y salidas) tiene la forma general:

```
                          "fmt" (var)
```

donde:

```
             (var)  :  indica a que corresponde esta parte en el
                       programa en C. No tiene por que ser una
                       variable. Tambien puede ser, por ejemplo,
                       una constante. Lo importante es que la
                       expresion entre parentesis se refiere a
                       la expresion de C que queremos meter en
                       el codigo en ensamblador.
             "fmt"  :  indica como se mete en el ensamblador la
                       expresion de C que hemos indicado con (var).
                       Hay muchas maneras de meter simbolos: como
                       constantes, como referencias absolutas de
                       memoria, como referencias de memoria inde-
                       xadas, como un registro arbitrario, como un
                       registro concreto, como un par de registros,
                       como un registro de la FPU, y algunas mas.
```

La forma del tercer campo (corrupto) es una lista de nombres de registros (entre comillas y sin el signo '%') que son corrompidos

como efecto colateral de nuestro codigo. Veremos ejemplos de corrupcion de registros en la seccion 3.3.5.
Asi pues, la forma mas general del comando __asm__ es:

```
         __asm__("..." ["..." ... ]
                 [ : ["fmt" (var) [, ...] ] ]
                 [ : ["fmt" (var) [, ...] ] ]
                 [ : ["reg" [, ...] ] ] );
```

Observa la logica y la simplicidad de la notacion. Como ya hemos indicado antes, hay que decirle a 'gcc' que pasa con las entradas, las salidas y la informacion corrupta. Pues bien, de esto se encarga cada uno de los tres campos. Ahora bien, para un dato de entrada, salida o corrupto, siempre tenemos que indicar al menos dos cosas: que forma tiene ese dato en el ensamblador y que forma tiene ese dato en el C. Una vez que el compilador sabe ambas cosas, no tiene mas que sustituir simbolos. De hacer esa doble identificacion se encargan los pares '"fmt" (var)'.

Bien, ahora sabemos la estructura de los tres campos, pero queda una cuestion por resolver. Dado que el ensamblador debe tener una idea clara de _absolutamente_todo_ dato que usamos en nuestro codigo ensamblador, no tiene sentido usar datos especificos en el codigo ensamblador.

Por ejemplo. Supongamos que queremos meter en la variable 'dato' (tipo long) la cantidad decimal 7. Si sabemos que el codigo almacena 'dato' en -4(%ebp), podriamos hacer:

```
        __asm__("  movl $7, -4(%ebp)" : "=m" (dato) );
```

y considerar que esto esta bien hecho. A fin de cuentas, hemos indicado al compilador que nuestro codigo tiene como dato de salida una referencia de memoria indexada (de ahi el "=m"), que va a caer en la variable 'dato'.

Pero esto es una metedura de pata. Lo que necesitamos es que el compilador _llene_ nuestro codigo con la posicion correcta de 'dato', no con la que nosotros le impongamos.

Esto implica que el codigo ensamblador debe ir escrito en forma de 'plantilla', de modo que el compilador lo llene con los datos correctos. Para esto se emplean los simbolos %0, %1, ... hasta %10 (hasta %30 en las versiones avanzadas de gcc).

Por ejemplo:

```
        __asm__(" movl $7, %0 " : "=m" (dato) );
```

es correcto, y le dice al compilador que coja la variable 'dato', la incruste en el codigo ensamblado en forma de acceso indexado a memoria en el lugar donde esta '%0', y que tenga en cuenta que 'dato' es variable de salida para este codigo.

Resumiendo: el formato __asm__ bien hecho esta compuesto de:

1) Tres campos en el formato arriba indicado, que indican los datos de entrada, salida y corruptos.
2) Un codigo ensamblador en forma de plantilla, que va entre comillas al principio del comando __asm__. Los 'huecos' de la plantilla se indican %0, %1, etc.

Una pregunta importante es como reconoce 'gcc' a que dato corresponden %0, %1, %2, etc. La cosa es facil: gcc va nombrando los datos por orden segun los va leyendo. Por ejemplo, en el codigo (que ya vimos en la seccion anterior):

```
            __asm__(" xorw %0, %0     \n\t" \
                    " xorw %1, %1     \n\t" \
                    " movw %2, %1     \n\t" \
                    " addw %3, %1     \n\t" \
                    " adcw %0, %0     \n\t"
                    : "=r" (acarreo) , "=r" (sum)
                    : "rm" (i) , "rm" (j) );
```

tenemos 4 datos entre entradas y salidas, es decir, que habra que definir simbolos desde %0 hasta %3. Entonces, tendremos que

%0 se corresponde con "=r" (acarreo), %1 con "=r" (sum),
%2 con "rm" (i) y %3 con "rm" (j). Se leen de izquierda a
derecha, segun van apareciendo.
    Si se entiende lo que hemos visto hasta ahora en esta seccion,
se tiene buena parte del camino andado. La idea general, resumida
una vez mas:
        1) Escribe el ensamblador usando simbolos, de manera que
           sea el compilador el que los rellene.
        2) Una vez que tienes tu ensamblador escrito con simbolos,
           escribe las lineas de entradas, salidas y corruptos
           de manera que esos simbolos sean sustituidos adecuada-
           mente (por registros, refs. a memoria, etc.) y se
           correspondan con los datos en C que te interesan.
Y eso es todo.
    Observa el ejemplo que acabamos de ver arriba. Para construirlo,
en primer lugar compuse el codigo en ensamblador usando simbolos.
Me quedo:

```
                    xorw %0, %0
                    xorw %1, %1
                    movw %2, %1
                    addw %3, %1
                    adcw %0, %0
```

Ahora, yo sabia que una instruccion tipo 'xor <algo>, <algo>'
solo puede ser sustituida correctamente si <algo> es un registro.
De manera que introduje en la primera linea de las salidas "=r",
que obliga a 'gcc' a sustituir %0 por un registro de su eleccion.
Ademas, como %0 debia ser al final del ensamblador el valor del
acarreo, le pedi a gcc que almacenase %0 en la variable 'acarreo',
lo cual se logra con poner (acarreo) despues del "=r". Con esto
ya podia estar seguro de que no tenia que preocuparme mas por %0.
Con %1, %2 y %3 procedi exactamente igual. El resultado final
fue el codigo completo, tal y como fue empleado en la seccion
anterior.
    3.3.1 Simbolos solapados.
    Aunque el formato expuesto hasta ahora es conceptualmente muy
sencillo, cuando uno echa las cosas a correr siempre aparecen
algunas complicaciones muy naturales que a primera vista no se
toman en cuenta.
    Piensa en el siguiente ejemplo: queremos hacer una diminuta
linea de ensamblador que calcule el 'not' (complemento a 1) de
una palabra. En principio podriamos hacer la construccion:

```
        __asm__("notl %0" : "=r" (var1) : "r" (var2));
```

Pero el problema que aparece es el siguiente: como el registro
de entrada es el mismo que el de salida, el compilador se va a
hacer un lio. Tal y como hemos escrito el codigo, 'gcc'
va a definir los simbolos %0 y %1, y se mostrara muy confundido
cuando vea que el codigo solo emplea %0.
    Una solucion aceptable seria hacer lo siguiente:

```
  __asm__("movl %0, %1 ; notl %1" : "=r" (var1) : "r" (var2));
```

Esto es perfectamente correcto, pero no siempre es aplicable (por
ejemplo, al usar la FPU no se tiene tanta facilidad para echar
mano de registros adicionales).
--> El problema es por tanto: como decirle a 'gcc' que un simbolo
--> de salida coincide con un simbolo de entrada.
    Naturalmente, esto tiene solucion; de hecho, ya hemos visto
algun ejemplo en las secciones anteriores. La solucion es simplemente
indicar en el argumento repetido el numero de la etiqueta que
se repite (sin el '%'). Por ejemplo:

```
        __asm__("notl %0" : "=r" (var1) : "0" (var2));
```

Con esto el compilador se da cuenta de que '%0' es compartido
por la entrada y la salida, dado que en la entrada aparece
"0" como formato, haciendo referencia a %0. Veamos algunos

ejemplos que empleamos anteriormente:

```
        __asm__("fsincos" : "=t" (x), "=u" (y) : "0" (t));
```

En este caso, los formatos "t" y "u" indican respectivamente la primera y la segunda posiciones en la pila de la FPU. El fragmento en ensamblador espera la entrada de 't' en la primera posicion de la pila y, tras la instruccion 'fsincos', se pone Sen(t) en la segunda y Cos(t) en la primera posiciones de la FPU. De manera que la variable de salida 'x' y la de entrada 't' comparten la primera posicion en la pila de la FPU. Por este motivo, se usa el "0" para indicar el solapamiento entre la entrada y la salida. (Este tipo de complicaciones es tipico de las operaciones con la FPU; como ejemplo es un tanto enrevesado).

Otro ejemplo de la misma naturaleza es la verion en ensamblador de 'y = exp(x)':

```
          __asm__(" fldl2e        \n\t" \
                  " fmulp         \n\t" \
                  " fld %0        \n\t" \
                  " fld %0        \n\t" \
                  " frndint       \n\t" \
                  " fsubrp        \n\t" \
                  " f2xm1         \n\t" \
                  " fld1          \n\t" \
                  " faddp         \n\t" \
                  " fscale        \n\t" \
                  " fxch          \n\t" \
                  " fstp %0       \n\t"
                  : "=t" (y)
                  : "0" (x) );
```

Una vez mas, la entrada y la salida comparten el registro %st(0) de la FPU, de manera que se usa el solapamiento. Veamos un ultimo ejemplo, tonto pero instructivo: un fragmento de ensamblador que calcula el complemento a 1 de una variable y el complemento a 2 de otra:

```
             __asm__(" notl %0    \n\t" \
                     " negl %1        "
                     : "=r" (var1) , "=r" (var2)
                     : "0"  (var3) , "1"  (var4) );
```

En este caso tenemos dos solapamientos, el de %0 y el de %1.

Como puedes ver, esta situacion ha sido cubierta con facilidad y elegancia por 'gcc'. Ningun problema por esta parte.

3.3.2 Formatos mas comunes.

Hasta ahora hemos recurrido a los ejemplos para aprender los formatos "=" (salida), "r" (registro), "m" (memoria indexada), "f" (registro de la FPU), "t" (primer registro de FPU), "u" (segundo registro de FPU), y los de solapamiento "0", "1", etc.

Tambien hemos observado (en 3.2) que se pueden combinar varias sugerencias en la misma cadena, como "rm" (registro o memoria indexada). Pueden hacerse muchas otras combinaciones, siempre que tengan sentido, como "=rm" o "mf", etc.

En la documentacion en ingles nuestros 'formatos' se traducen como 'constraints' (restricciones). Para informarse sobre todos los tipos importantes, incluyendo algunos a medio documentar (como los registros SSE), consulta la documentacion de referencia del 'gcc'.

FORMATOS GENERICOS DE LA CPU:

"r" : cualquier registro de la CPU (%eax, %ebx, etc.)
"m" : una referencia a memoria (p. ej. -8(%eax, %ebx, 2)).
"q" : un registro que que puede operar con bytes.
Es decir, %eax, %ebx, %ecx o %edx.
"A" : el par de 64 bits %edx:%eax. Tambien vale para indicar uno cualquiera de los dos, %edx o %eax.

```
    REGISTROS ESPECIFICOS DE LA CPU:
    "a"  :  %eax, %ax, %ah, %al.
    "b"  :  %ebx, %bx, %bh, %bl.
    "c"  :  %ecx, %cx, %ch, %cl.
    "d"  :  %edx, %dx, %dh, %dl.
    "D"  :  %edi, %di.
    "S"  :  %esi, %si.

    CONSTANTES ENTERAS:
    "i"  :  una cantidad inmediata de 32 bits.
    "n"  :  una cantidad inmediata conocida en tiempo de
            compilacion.
    "I"  :  una constante entera entre 0 y 31. Se usa para
            indicar desplazamientos (shl, ror, etc.)
    "J"  :  una constante entera entre 0 y 64. Se usa para
            desplazamientos de 64 bits.
    "K"  :  equivale a 0xff.
    "L"  :  equivale a 0xffff.
    "M"  :  0, 1, 2 o 3. Util para el factor de escala en 'lea', o
            en cualquier direccionamiento indexado.
    "N"  :  constante entera entre 0 y 255.
    REGISTROS DE LA FPU:
    "f"  :  cualquier registro de la FPU.
    "t"  :  el TOS ('top of stack') de la FPU: %st(0).
    "u"  :  %st(1).
    "G"  :  una constante en doble precision estandar para el 387.
    "F"  :  una constante en doble precision.

    COSAS CURIOSAS:
    "x"  :  un registro MMX.
    "y"  :  un registro SSE.
Gracias a estos formatos podemos especificar mediante simbolos
practicamente cualquier cosa. En la siguiente seccion veremos
ejemplos que se corresponden con los mas interesantes de estos
formatos, pero antes veamos otro tipo de formatos, de los que
forma parte el "=" que vemos tan a menudo en los argumentos de
salida:
    MODIFICADORES:
    "="  :  cuando se indica con un formato, se da a entender que
            el simbolo es escrito pero no leido por el codigo en
            ensamblador. Es comun usar "=" como formato para
            cualquier salida, sobre todo en versiones mas viejas
            de gcc.
    "+"  :  esto indica que el argumento es leido y escrito por
            el codigo. Es mas restrictivo que "=", dado que implica
            que 'gcc' no puede utilizar este simbolo como almacena-
            miento intermedio antes de ser escrito de manera
            definitiva. Me parece que el compilador se porta de
            manera caprichosa con esta opcion; es casi mejor pasar
            de ella. Aun asi, en ocasiones funciona.
Existen otros modificadores, pero corresponden a casos de mucha
mas sutileza, y no son criticos. Observa la documentacion para
mas detalles.
    3.3.3  Ejemplos de formatos de la CPU.
    Ya hemos visto ejemplos del uso de "r", "m", "A", "=", "f",
"t" y "u". Tambien de algunas combinaciones, como "=r" y "rm".
Acerca de la FPU hablaremos con mas detalle en 3.3.4, asi que
por ahora nos concentraremos en la CPU.
    En primer lugar, veamos que pasa con las operaciones a nivel
de byte. Si queremos usar registros que funcionen con bytes es
importante que nos limitemos a "q", "a", "b", "c" o "d". Por
ejemplo:
```

```
        __asm__("negb %0" : "=q" (var1) : "0" (var2));
        __asm__("negb %0" : "=a" (var1) : "0" (var2));
```

curiosamente, he intentado compilar infringiendo la regla, con:

```
        __asm__("negb %0" : "=S" (var1) : "0" (var2));
```

y el compilador, sabiamente, ha cambiado %esi por %al de manera automatica. Aun asi es mejor no tentar a la suerte.

Veamos un uso de constantes raras. Mete esto en un codigo en C:

```
        #define FACT 2
        main()
        {
            long var1, var2;

            __asm__(" shll %1, %0 "
                    : "=a" (var1)
                    :  "I" (FACT), "0" (var2) );
        }
```

Si lo compilas con ’-S’ y te fijas en la zona injertada, tenemos:

```
        /APP
                shll $2, %eax
        /NO_APP
```

que es justo lo que le pedimos. Esto es un buen ejemplo del uso de un formato raro como "I". De paso te permite ver como se pueden meter definiciones del preprocesador en el ensamblado inline.

Vamos a echar una mirada a las constantes inmediatas. Prueba a compilar con ’gcc -S’ el siguiente codigo:

```
        #define BASE 23144
        main()
        {
            long var1;
            __asm__("movl %1, %0" : "=m" (var1) : "i" (BASE + 69));
        }
```

compliando y editando el ensamblador obtenemos:

```
        /APP
                movl $23213, -21(%ebp)
        /NO_APP
```

Esto da un ejemplo de cantidad inmediata de 32 bits, y de operaciones dentro de los valores introducidos en el ensamblador, mezclado con preprocesamiento. Puedes desarrollar ejemplos mucho mas complejos.

Llegados a este punto, supongo que te haces una idea de como se manipula el resto de los formatos. Sencillamente, siempre hay uno (o quizas varios) formatos adecuados para una cierta situacion. Uno debe ser lo menos restrictivo posible con el tipo de formato, para no contrarrestar la capacidad de optimacion del ’gcc’.

3.3.4 Ejemplos de formatos de la FPU.

La FPU de los x86 es bastante curiosa. Si llevas algun tiempo en el mundo de la informatica, es posible que recuerdes FORTH, un genial lenguaje interpretado que empleaba varias pilas en sus accesos a la memoria, en vez de accesos aleatorios a traves de variables. La FPU de los x86 fue concebida de la misma manera, lo que la hace francamente entretenida de programar. Quizas un fallo de los lenguajes tipo FORTH es que son poco concurrentes, dado que es muy dificil paralelizar instrucciones que acceden a una pila.

Lo que nos ocupa ahora es que la estructura de pila de la FPU hace muy dificil el andar jugando con registros especificos de la FPU. Mientras que con la CPU podiamos especificar alegremente que registro de proposito general queriamos usar, esto no es factible con la FPU (salvando %st(0) y %st(1) que, como vimos, se pueden indicar con los formatos "t" y "u"). Sin embargo, en muchas situaciones es suficiente con poder usar %st(0) y %st(1), si se programa con cuidado.

Ensamblar inline con la FPU se parece un poco a lo que haciamos

en la Parte 2 con las subrutinas. En los argumentos de entrada le decimos al 'gcc' que datos queremos que nos meta en la pila, mientras que en las salidas le decimos que datos hemos dejado en la pila para que los recoja.

Un ejemplo lo tenemos en el caso que ya hemos visto:

```
__asm__("fsincos" : "=t" (x), "=u" (y) : "0" (t));
```

El dato de entrada, "0" (t), le dice al compilador que nuestro codigo en ensamblador asume que en %st(0) se encuentra la variable 't'. Entonces, 'fsincos' toma de la pila este valor y devuelve Cos(t) en %st(0) y Sen(t) en %st(1). Los argumentos de salida le dicen al compilador que puede recoger de esas dos posiciones de la pila los valores de 'x' e 'y'.

El otro caso que hemos visto, el de la exponencial, es tambien interesante:

```
__asm__(" fldl2e        \n\t" \
        " fmulp         \n\t" \
        " fld %0        \n\t" \
        " fld %0        \n\t" \
        " frndint       \n\t" \
        " fsubrp        \n\t" \
        " f2xm1         \n\t" \
        " fld1          \n\t" \
        " faddp         \n\t" \
        " fscale        \n\t" \
        " fxch          \n\t" \
        " fstp %0       \n\t"
        : "=t" (y)
        : "0" (x) );
```

Una vez mas, el dato de entrada es 'pusheado' por el compilador, que recoge de la pila el dato de salida una vez que hemos terminado. Lo interesante de este ejemplo es que durante su ejecucion se introducen y se sacan muchos otros datos de la pila (como el logaritmo de 'e' en base 2) pero, debido a que todos los datos adicionales que se introducen son sacados antes del final del injerto, el 'gcc' no tiene por que preocuparse de eso.

Esto es un principio fundamental. Si te ocupas de mantener la coherencia de la pila, y te limitas a decirle al 'gcc' que meta o saque los datos que necesitas, escribir codigo para la FPU es algo realmente facil.

De todos modos, hay sutilezas que debes tener en cuenta para que el compilador se entere de que manipulaciones has operado en la pila. Si estas interesado en escribir ejemplos complicados de ensamblador inline con manipulaciones en la pila un tanto exoticas (por ejemplo, cuando hay mas valores de entrada que de salida), debes consultar la documentacion de 'gcc' (parte 'Assembler Instructions with C Expression Operands') que ofrece una lista de pasos a seguir para manejar esos casos curiosos. El problema esta en que el compilador puede tener problemas para decidir si los datos que se introdujeron inicialmente en la pila fueron expulsados de ella o no por tu codigo en ensamblador. Ese tipo de circunstancias es facil de decidir, aunque introducir las reglas en un tutorial como este podria ser un tanto molesto para los que no pretenden dedicarse especificamente a programar la FPU.

Mi consejo es que uses ensambles inline solo instrucciones de la FPU relativamente cortas pero exoticas, como las que hemos visto, en vez de escribir partes largas que te llevaran a complicaciones innecesarias (aunque no insalvables, en absoluto).

3.3.5 Registros corruptos.

Hemos estado obviando este campo de la instruccion __asm__ hasta ahora, no porque sea complicado (no lo es en absoluto), si no porque no es muy usual encontrarse con este tipo de casos.

Veamos sin embargo un primer ejemplo: supongamos que queremos hacer la instruccion 'rdtsc', pero almacenando solo los 32 bits menos significativos del par %edx:%eax, en tanto que nos importa poco que sucede con los bits mas significativos. En otras palabras, nos quedaremos con el contenido de %eax, despreciando lo que haya en %edx (esta situacion puede aparecerte generando numeros aleatorios mediante temporizadores de mucha precision).

En este caso, tenemos que advertir a 'gcc' que el registro %edx, aunque no forma parte de nuestras entradas ni salidas, ha sido alterado. Esto se indicaria como:

```
        __asm__(".byte 0x0f, 0x31"
                : "=a" (tiempo)
                :
                : "edx" );
```

El compilador asume facilmente la situacion.

Observa que los registros corruptos van, como ya indicamos en 3.3, con su nombre completo y sin el '%' de prefijo.

Vamos a ver un caso mucho mas divertido. Ejecutemos la instruccion 'cpuid' y extraigamos la familia, modelo y 'stepping' del micro que estamos empleando. Para ello hay que ejecutar las instrucciones en ensamblador:

```
          movl $1, %eax
          cpuid
```

Despues de estos comandos se tiene que %eax almacena los datos que queremos, en tanto que %ebx y %ecx quedan en estado indefinido, y %edx contiene informacion sobre las facilidades que ofrece el microprocesador. Aqui tenemos casos sobrados de corrupcion. Vamos a hacer un programilla que nos diga algo sobre el micro. Guarda el siguiente codigo en un fichero ('cpuid.c'):

```
        #include <stdio.h>
        main()
        {
          unsigned long id, carac;
          /* Esto hace el cpuid */
          __asm__(" movl %2, %0        \n\t" \
                  " .byte 0x0f, 0xa2   "
                  : "=a" (id) , "=d" (carac)
                  : "i" (1)
                  : "ebx" , "ecx" );
          /* Ahora escribimos algo de informacion. */
          printf("Informacion del microprocesador:\n");
          printf("    Familia : %d\n", ((id & 0xf00)>>8));
          printf("    Modelo  : %d\n", ((id & 0xf0)>>4));
          printf("    Stepping: %d\n\n", (id &0xf));
          /* Ahora deducimos alguna facilidad del micro. */
          if (carac & (1<<23))
              printf("El micro soporta MMX\n");
          else
              printf("El micro no soporta MMX\n");
          if (carac & (1<<15))
              printf("El micro soporta CMOV\n");
          else
              printf("El micro no soporta CMOV\n");
          if (carac & (1<<5))
              printf("El micro soporta MSRs\n");
          else
              printf("El micro no soporta MSRs\n");
          if (carac & (1<<25))
              printf("El micro soporta XMM\n");
          else
              printf("El micro no soporta XMM\n");
          exit(0);
```

```
        }
Compilamos y ejecutamos:
        demeter# gcc -o cpuid cpuid.c
        demeter# ./cpuid
        Informacion del microprocesador:
            Familia : 6
            Modelo  : 2
            Stepping: 1
        El micro soporta MMX
        El micro soporta CMOV
        El micro soporta MSRs
        El micro no soporta XMM
        demeter#
Lo cual es correcto. Mi micro es uno de los primeros AMD K7, de
ahi que tenga un Modelo y Stepping tan bajos. Naturalmente,
soporta MMX, CMOV y MSRs (Model Specific Registers). Pero, al
ser un AMD, no soporta la basurilla multimedia de Intel; tiene
su propia basurilla multimedia (3DNow! y todo eso). Desarrollando
un poco este codigo se podria lograr un buen identificador de
micros.
---| 3.4  Que se nos queda en el tintero.
    Intentar cubrir todas las opciones del ensamblado inline es
un proyecto enormemente ambicioso, que no se podria conseguir sin
complicar mucho este tutorial. Creo que con lo que hemos visto
hasta ahora tienes, como minimo, para cubrir todos los casos
importantes con los que te puedes topar en la practica.
    Al menos, espero haber podido dar en esta parte los funda-
mentos para entender otros textos que circulan en la red (todos los
que he visto estaban en ingles salvo uno, en italiano) sobre
ensamblado inline. Es sorprendente lo facil que es escribir
ensamblador inline una vez que uno se familiariza un minimo con
la materia.
    Como ya he indicado, nos falta por detallar algunas opciones
especiales en los formatos de entrada y salida (cosas como "%",
"&", "#", etc.) Tambien nos hemos dejado en el tintero el mecanismo
para tratar el ensamblado inline de la FPU en sus formas mas
complicadas, pero eso es algo que puedes consultar en la
documentacion 'gnu' una vez que tienes una experiencia minima.
    Seguro que quedan otros muchos agujeros por tapar, algunos de
ellos imperdonables. Si encuentras alguno, y crees que debe taparse,
enviame un mail. Si crees que puedes hacerlo tu mismo, adelante,
este documento es _tu_ documento. Disfrutalo.
----------------------------| Referencias |----------------------------
He aqui lo que puedes consultar, antes, despues o durante la lectura
de este tutorial. La bibliografia no es completa, pero puede ayudarte
a conseguir una buena cultura elemental en el tema del ensamblador
AT&T con 'gcc'. Me he tomado la libertad de comentar algunas de las
referencias.
---| Jon Beltran de Heredia, "Lenguaje Ensamblador de los 80x86"
     Anaya Multimedia, ISBN 84-7614-622-1
    Desde mi punto de vista, la mejor introduccion al ensamblador
x86 en nuestra lengua. Aunque solo cubre el ensamblado en 16 bits
en formato Intel (para el anticuado 'masm'), es insustituible si
quieres aprender los fundamentos del ensamblador en estas maquinas.
---| Intel Corp., "Intel Architecture Software Developer's Manual.
                   Volume 2: Instruction Set Reference".
     Order number 243191.
    El 'white paper' sobre la arquitectura x86 por excelencia. Los
volumenes 1 y 3 (Order # 243190 y 243192 respectivamente) son tambien
muy interesantes. Si quieres conseguirtelos en .pdf vete a la web
de Intel y busca el documento. Te recomiendo que lo hagas por el
'Order number', que es la manera mas rapida de dar con el. La pagina
```

web de AMD tambien tiene ’white papers’ muy interesantes.
---| GNU, "Using the GNU Compiler Collection"
    Este documento cubre todos los detalles relevantes del compilador ’gcc’, y puedes descargartelo desde www.gnu.org en varios formatos. Mucha de la informacion que hemos omitido por ser latoso o complicado incluirla aqui puedes encontrarla en este documento.
---| GNU, "The GNU Assembler"
    Otro documento de www.gnu.org. Este documento es insustituible para enterarse de las directivas y principales convenios del funcionamiento del compilador ’gas’. Siempre es bueno tenerlo a mano.
---| Colin Plumb, "A Brief Tutorial on GCC inline asm (x86 biased)"
     20 April 1998.
    Una de esas joyas con las que te encuentras en la red. Escrito por el hacker Colin Plumb, es una fantastica introduccion al ensamblado inline, publicado originalmente en una lista de correo y actualmente disponible en varias URLs (una busqueda en google deberia localizartelo). Yo encontre una copia en www.opennet.ru, en una coleccion de documentos sobre ensamblador. Absolutamente imprescindible.
---| Aparte de estos documentos puedes encontrar muchas otras referencias, buena parte de ellas obras amateur con mas buena intencion que rigor. De todos modos, un google con ’inline assembler’ arroja muchos resultados, algunos de ellos de primera linea.
EOF

```
-[ 0x03 ]----------------------------------------------------------------
-[ Bazar ]----------------------------------------------------------------
-[ by Varios autores ]---------------------------------------------SET-29--
```

Otro numero mas donde damos la oportunidad de publicar en SET a gente que no se ve con animos para escribir penyazos tan largos como los que suelen ir sueltos o quieren enviarnos sus trucos, opiniones o pequenyos descubrimientos.

De todas manereras, ya sabeis que lo que nos gusta es que os mojeis y hagais vuestros articulos cuanto mas completos mejor!

```
-----------------[  Contenidos del Bazar de SET 29  ]-----------------------
```



```
-[ 3x01 ]----------------------------------------------------------------
-[ RompeCabezas ]--------------------------------------------------------
-[ by Lindir ]-----------------------------------------------------SET-29--
```

Me consta que muchos lectores de SET se dedican a la programacion. Algunos lo hacen por hobby y otros por trabajo. Pero lo que propongo aqui es divertirnos un poco con lo que podria llamarse "programacion limitada".

Hoy en dia que tan de moda estan los deportes de riesgo, vamos a intentar el mas dificil todavia de la programacion en C: acercarnos al limite del lenguaje y el compilador. No os asusteis. No soy ningun guru y posiblemente muchos de vosotros resolvais los pasatiempos enseguida, aunque si estais empezando a programar en C puede que os cuesten bastante.

Propongo a continuacion ocho acertijos de programacion en lenguaje C. No se trata de realizar programas largos o complicados, sino de realizar programas triviales si no fuera por este detalle: tendremos la sintaxis limitada.

Cada acertijo tiene su propias reglas, que pueden incluir el no utilizar ciertas palabras reservadas (for, switch ...), usar solo una cantidad determinada de variables, etc. Ademas hay ciertas reglas comunes: no pueden usarse funciones de la biblioteca estandar (salvo printf y scanf) ni enlazar el codigo con ningun modulo externo. Los programas deben constar de un solo archivo C. Aparte de esto, todo lo que no se indique expresamente como prohibido estara permitido. Por supuesto, esta expresamente prohibido el uso de ensamblador en linea o utilizar buffers de codigo: todo debe hacerse en C.

Como guia, los acertijos estan ordenados, y los primeros pueden resolverse de forma mas sencilla que los ultimos, en general. Ademas algunos estan relacionados con los siguientes. Por lo tanto, si sois capaces de resolver los ultimos podreis utilizar lo aprendido para resolver los primeros de forma distinta a la solucion propuesta.

Tras cada uno de estos pasatiempos incluyo una solucion al mismo. La solucion no tiene por que ser unica ni la mejor posible. He intentado que sea corta cuando esto ha tenido sentido, y que sea clara en la mayor parte de los casos, incluyendo comentarios al codigo.

Notar que las soluciones pueden no funcionar en ciertos compiladores o maquinas. Su correccion esta comprobada usando el gcc 2.95.3 (si, es muy antiguo) bajo Linux en un x86. Seguro que versiones mas modernas del mismo compilador tambien sirven. Tambien puede ser que segun las opciones de compilacion (alineacion de la pila, etc.) alguna no funcione. Como pista, yo utilizo la orden "gcc archivo.c -o archivo". He intentado evitar que el compilador muestre "warnings", salvo el de que la funcion main devuelve void en lugar de int.

Estas soluciones propuestas estan escritas "al reves". Cada linea debe ser leida de derecha a izquierda. Incluyo un peque¤o programa que "da la vuelta" a las lineas recibidas por la entrada estandar, para que podais comprobar de forma mas rapida si vuestra solucion coincide con la propuesta.

Os aconsejo que si alguno no os sale a la primera lo penseis un tiempo y no mireis la solucion enseguida, puesto que ya no tendra gracia volverlo a intentar. En algun caso el siguiente pasatiempo puede daros una pista de como resolver el anterior.

-Vuelta.c-

```
/* vuelta.c: programa que toma lineas de la entrada estandar y las escribe
caracter a caracter desde el final al principio por la salida estandar. */
#include <stdio.h>
void main (void){
        char linea[256];
        int c,n=0;
        while ((c=getchar())>0){
                if (c=='\n'){
                        int i;
                        for (i=n;i>0;i--)
                                putchar(linea[i-1]);
                        putchar(c);
                        n=0;
                }
                else linea[n++]=(char)c;
        }
}
/*************************** fin de vuelta.c ******************************/
```

-Acertijos-

```
/*************************************************************************
1) Pasatiempo en C n§ 1:

        Sin utilizar if, do, while, for, switch, goto ni el operador ternario
        ?, hay que realizar un programa que compare dos numeros enteros (a y
        b) que reciba por teclado e imprima un mensaje en la pantalla
        indicando cu l de ellos es mayor ("a es mayor/menor que b").
        S¢lo se permite llamar a scanf() y printf() para recibir los n£meros
        e imprimir los mensajes. No se permite llamar a ninguna otra funci¢n
        de biblioteca.
*/


/* Soluci¢n: */
>h.oidts< edulcni#

{)diov(niam diov
;b,a tni
;)b&,a&,"d% d%"(fnacs
```

```
;)b,a,"n\d%>d%"(ftnirp && ))b,a,"n\d%<d%"(ftnirp! || b>a(
}
/*************************************************************************/

/*************************************************************************
2) Pasatiempo en C n§ 2:

        Se trata de, sin usar mas constantes que las cadenas pasadas a
        printf y scanf para recibir los datos e imprimir los mensajes y sin
        usar operadores *, /, ni ++, hacer un programa que multiplique dos
        numeros que reciba por teclado e imprima un mensaje con el resultado
        de la operacion.
        No puede usarse ningun tipo de constante literal, ni siquiera
        constantes numericas  (1,2...) o de caracter ('a','\0'...).
*/

/* Solucion: */
>h.oidts< edulcni#

{)diov(niam diov
;d,c,b,a dengisnu
/* ;1 = d */                    ;)a&,"d%"(fnacs=d
;)b&,"d%"(fnacs
/* 0 = c */                                 ;c=^c
/* rodacilpitlum elcub */                   };b=+c;d=-a{)a(elihw
;)c,"n\d%"(ftnirp
}
/*************************************************************************/

/*************************************************************************
Pasatiempo en C n§ 3:

        Al igual que el anterior, se trata de, sin usar mas constantes que
        las cadenas pasadas a printf y scanf para recibir los datos e imprimir
        los mensajes y sin usar operadores *, /, ni ++, hacer un programa que
        multiplique dos numeros que reciba por teclado e imprima un mensaje
        con el resultado de la operacion.
        No es valido usar los valores de retorno de las funciones de stdio.h.
*/

/* Solucion: */
>h.oidts< edulcni#

{)diov(niam diov
/* tesffo elpmis nU */                      ;f,e rahc
/* 1 = d */             ;)f&(-)e&(=d,c,b,a dengisnu
;)b&,a&,"d% d%"(fnacs
/* 0 = c */                                 ;c=^c
/* rodacilpitlum elcub */                   } ;b=+c ;d=-a {)a(elihw
;)c,"n\d%"(ftnirp
}
/*************************************************************************/

/*************************************************************************
Pasatiempo en C n§ 4:

        Utilizando solo una variable, una llamada a printf, y el caracter
        ; dos veces, hay que escribir un programa que imprima los numeros del
        1 al 5, sin utilizar ninguna constante numerica ni llamada a funcion.

*/
```

```
/* Solucion: */
>h.oidts< edulcni#

{)diov(niam diov
;c^c=c tni
;)c++,c++,c++,c++,c++,"n\d% d% d% d% d%"(ftnirp
}
/*************************************************************************/

/*************************************************************************
Pasatiempo en C n§ 5:

        Usando el caracter ; solo dos veces y unicamente dos variables,
        escribir un programa que multiplique dos numeros que reciba por
        teclado.
*/

/* Solucion: */
>h.oidts< edulcni#

{)diov(niam diov
;b,a tni
;))b&,a&,"d% d%"(fnacs,b*a,"n\d% :se odatluser lE"(ftnirp
}
/*************************************************************************/


/*************************************************************************
Pasatiempo en C n§ 6:

        Sin utilizar operadores de multiplicacion ni bucles for/while/do
        y usando el caracter ; solo tres veces, escribir un programa que
        multiplique dos numeros unsigned >0 que reciba por teclado.
*/

/* Solucion: */
>h.oidts< edulcni#

/* noicacilpitlum ed ovisrucer omtiroglA */
{)b dengisnu ,a dengisnu(itlum dengisnu
;)b,1-a(itlum+b:b?1==a nruter
}

{)diov(niam diov
;b,a dengisnu
;))b&,a&,"d% d%"(fnacs,)b,a(itlum,"n\d% :se odatluser lE"(ftnirp
}
/*************************************************************************/

/*************************************************************************
Pasatiempo en C n§ 7:

        Sin utilizar el caracter ; ni ninguna constante mas que las cadenas
        de formato para printf/scanf, escribir un programa que reciba un
        numero (se supone > 0) por teclado y escriba todos los numeros desde
        este al 1.
*/


/* Solucion: */
>h.oidts< edulcni#
```

```
{)vgra** rahc ,cgra tni(niam diov
}{)))cgra&,"d%"(fnacs!(||)cgra=^cgra((elihw
)cgra(elihw
}{)cgra--||)cgra," d%"(ftnirp!(elihw
}
/*************************************************************************/

/*************************************************************************
Pasatiempo en C n§ 8:

        Sin utilizar el caracter ; ni los operadores * / ni mas constantes
        que las cadenas de formato de printf/scanf, escribir un programa
        que reciba dos numeros (se suponen > 0) por teclado e imprima el
        resultado de multiplicarlos.
*/


/* Solucion: */
>h.oidts< edulcni#

redop arap eneit es olos ortemarap omitlu lE .noicacilpitlum ed noicnuF */
/* ; le razilitu nis sam lacol elbairav anu noc ratnoc
{)c tni ,b* tni ,a* tni(itlum tni
/* ;0=c */                      }{)c=^c(fi
/* rodacilpitlum elcub */                       {)a*(elihw
}{)--)a*((fi
}{))b*(=+c(fi
}
/* aicnerefer rop odatluser led osaP */         }{)c=a*(fi
}

{)vgra** rahc ,cgra tni(niam diov
/* ;0=vgra=cgra ,oremirP */
        }{))vgra)tni(^vgra)tni(=vgra)tni(((||)cgra^cgra=cgra((fi

/* odalcet rop adartnE */
        }{))vgra&,cgra&,"d% d%"(fnacs!(fi

/* .odatluser led noiserpmi e noicacilpitluM */
}{))cgra^cgra(&&)cgra,vgra&)* tni(,cgra&(itlum(fi
 }{))cgra,"n\d%"(ftnirp!(fi
}
/*************************************************************************/
```

```
-[ 3x02 ]-------------------------------------------------------------
-[ Mini ejemplo ]-----------------------------------------------------
-[ by Astaroth H ]-----------------------------------------SET-29--
```

"El mar siempre esta ahi y es imposible aprender a nadar si nunca intentamos sumergirnos en el".

(Relis).

```
 ###########....___Introduccion___....#############
 Bueno este es mi primer tutorial, espero que lo entiendan.
 El programa con que comienzo es interesante a parte de que es un buen
```

explorador de windows de 1 a 10 le pongo 7, es interesante porque voy a utilizar varias herramientas, y para que uno que no sabe utilizarlas las aprenda facilmente, entonces a por el perrito.

```
Programa    : Super Explorer  v 1.4
Proteccion  : Tiempo limite de 15 dias y esta empaquetado.
Descripcion : Es un buen explorador de windows.
Dificultad  : Newbie.
Web site    : http://www.galcott.com
Herramientas: Exescope v 6.30
              ProcDump32 v 1.6
              W32Dasm v 8.93
              Hex Workshop v3.02
              (Las pueden conseguir de cualquier lado hay muchas).
Cracker     : Cracker

##########=....___Comenzando___....==========
```

Primero: (A jugar perrito).
Lo instalamos, lo ejecutamos y una nag nos dice que este programa no esta registrado, luego entramos y miramos que de bueno tiene el programa, jugamos con el un rato y a mi me parecio interesante, y hay muchas que hacer con el, pero bueno a crackearlo, para eso primero vamos al menu Help y escogemos Enter Registration Code, nos sale una ventana solo con un espacio para poner nuestro codigo ponemos cualquiera y OK, y un mensaje diciendo "Incorrect Registration Code", lo anotamos y salimos del programa.

Segundo: (Ahora la patita, la patita perrito).
Siempre antes de crackear hay que analizar el programa, por ahi hay muchas herramientas yo utilizo el Exescope v 6.30 , entonces ejecutamos Exescope y abrimos superexp.exe y nos aparece tres menus: Header (informacion del programa), Import (las funciones que importa) y Resource (Recursos del programa).

Entramos a Header y vamos a encontrar varios submenus (No voy a ahondar en todo lo que significa esto, si quieren mas informacion de esto, busquen por ahi tutoriales sobre encabezados PE) , y nos dirigimos a Section Header y vamos a ver que hay UPX0, UPX1 y .rsrc, esto significa que esta empaquetado con un uno que se llama UPX; ahora que sabemos que esta empaquetado hay que desempaquetarlo. (Si desean pueden ver lo que hay en los otros menus y submenus).

Tercero: (Hazte el muerto perrito).
Para desempaquetarlo voy a utilizar ProcDump32 v 1.6, ejecutamos ProcDump32 y dentro nos dirigimos a Unpack y escogemos UPX (el empaquetador) damos a OK y luego abrimos superexp.exe, y nos sale un mensaje le damos a aceptar, esperamos un momento y nos sale una pantalla diciendonos con que nombre vamos a guardarlo, ponen el nombre que les guste y guardar, despues aparece un mensaje diciendonos que a terminado, le damos a aceptar y listo programa desempaquetado, ahora podemos trabajar con el desensamblador. (Como observacion el original pesa 905KB y el desempaquetado 2,854KB).

Cuarto: (Vamos perrito, traeme el palo perrito).
Vamos a desensamblarlo, (Como observacion si intentamos desensamblarlo sin desenpaquetarlo nos saldra un mensaje diciendo que el W32Dasm a efectuado una operacion no admitida y que se tendra que cerrar). Ejecutamos W32Dasm v8.93 y abrimos el desempaquetado y podemos ir a preparar un cafe, porque se va a demorar un par de minutos.

Una vez dentro buscamos en las SDR (String Data References), la cadena que anotamos al comienzo la del mensaje del error "Incorrect Registration Code", cuando la encontremos hacemos doble click y nos lleva a una parte del codigo

le damos otra vez doble click a la cadena para saber si hay mas
coincidencias en el codigo pero solo hay una que es esta:

```
:005DBF53 BAE4BF5D00            mov edx, 005DBFE4
* Possible StringData Ref from Data Obj ->"You have been successfully
                                         registered."
|
:005DBF58 B8F4BF5D00            mov eax, 005DBFF4
:005DBF5D E87A91FEFF            call 005C50DC
:005DBF62 A178D26200            mov eax, dword ptr [0062D278]
:005DBF67 8B00                  mov eax, dword ptr [eax]
:005DBF69 E88AB9E7FF            call 004578F8
:005DBF6E EB14                  jmp 005DBF84   ; Salto incondicional que
                                                 nos evita el mensaje de error
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:005DBEB3(C)
|
:005DBF70 B930000000            mov ecx, 00000030

* Possible StringData Ref from Data Obj ->"Error"
|
:005DBF75 BA94C05D00            mov edx, 005DC094

* Possible StringData Ref from Data Obj ->"Incorrect registration code"
|
:005DBF7A B8A4C05D00            mov eax, 005DC0A4           ; Aparecemos
                                                              aqui
:005DBF7F E85891FEFF            call 005C50DC

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:005DBF6E(U)
|
:005DBF84 33C0                  xor eax, eax
```

Un salto condicional nos trae aqui y un poco mas arriba hay un salto
incondicional que evitaria mostrar el mensaje de error y arriba de ese salto
esta el mensaje de felicitacion por registrarnos, pero miremos quien nos
trae al mensaje de error, asi que busquemos con la linterna la direccion
005DBEB3 , nos lleva aqui:

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:005DBE44(C)
|
:005DBEB3 0F85B7000000          jne 005DBF70    ; Aparecemos aqui
:005DBEB9 B201                  mov dl, 01
:005DBEBB A1400B4800            mov eax, dword ptr [00480B40]
:005DBEC0 E8974EEAFF            call 00480D5C
:005DBEC5 8BD8                  mov ebx, eax
```

Miremos que tambien se viene aqui con un salto condicional y cuando llega
esta el salto que si no son iguales, no se que cosa, salta y el error, y si
son iguales nos muestra el mensaje de felicitacion asi que podemos nopear
toda esa instruccion osea:

```
cambiar :    005DBEB3 0F85B7000000        jne 005DBF70
por     :    005DBEB3 909090909090        nop nop nop nop nop nop
```

Los 90 significa NOP’S en hexadecimal, nop significa que no hace nada y asi
siempre nos muestra el mensaje de felicitacion .

Para eso ponemos el seleccionador (marca celeste) del W32Dasm encima de la
instruccion

```
: 005DBEB3 0F85B7000000          jne 005DBF70
```
y en la parte de abajo miramos el offset de esa instruccion la anotamos, y luego entramos en el Hex Workshop abrimos el ejecutable buscamos el offset y cambiamos los bytes y lo guardamos.

Luego ejecutamos Super Explorer introducimos cualquier codigo y listo programa registrado.
(Eso perrito asi se hace).

Observacion: En el desensamblado si subimos un poco por encima del mensaje bueno, veremos que entra en el registro para meter la clave que hayamos escrito.

Comentarios a astaroth_h@hotmail.com

Chau y espero que les haya gustado este tutorial, y espero que no sea el ultimo que escriba.

```
-[ 3x03 ]-------------------------------------------------------------------
-[ Introduccion a los mainframes ]-------------------------------------------
-[ by Anonimo ]-------------------------------------------------SET-29--
```

1. Prefacio

Vaya hombre... una busqueda del termino ’mainframe’ en todos los numeros de SET y ni una sola referencia. Con la ilusion que me hacia... ¨Sera que no interesa el tema? Solo dire que los bancos son de los principales usuarios; eso ya los hace muy atractivos de estudiar. ¨Estaran pasados de moda? Pues tampoco, aunque en los 80 parecia que iban a desaparecer.

En un primer momento pense que lo que pasaba es que los mainframes son bastante desconocidos y que ademas los hackers no hablaban de ellos en las e-zines porque les da rabia la imposibilidad para hackearlos.

En realidad hice mal la busqueda. Hay referencias a ’Main frames’ en SET15 (Red Global de IBM, de fca0000, y en los SET24 (Modems, de IMOEN) y SET04 (Metodos de hacking, de warezzman!!!). Que bien... pero no me satisfacen. -Yo quiero mas informacion!

El presente texto pretende ofrecer una vision global del confuso concepto de mainframe.

La informacion aqui contenida no siempre es correcta ni objetiva (menos da una piedra).

2. Que es un mainframe?

Eso me pregunto yo. Despues de dudar de mi propia idea de mainframe, y tras buscar en el Google, llego a la conclusion de que en Internet hay demasiada informacion y ningun metodo automatico para resumirla al gusto de uno mismo.

Asi que he intentado resumir las referencias del apartado final (y otras que no menciono). Y este es el resultado.

Etimologicamente, mainframe se refiere a la estructura o sistema (frame) principal (main), en referencia a la convergencia en un solo bloque principal de las diferentes unidades (procesamiento, comunicacion, datos...) que tuvo lugar con los primeros mainframes.

El concepto suele aplicarse sobretodo a ordenadores grandes en general, variando entre miniordenadores y macroordenadores segun la definicion. Los originales ocupaban grandes salas. Los actuales son como armarios. Otras definiciones no menos precisas, se concentran en el hecho de procesar muchos datos, o procesar datos muy deprisa, o lo equiparan al concepto de servidor grande o al concepto de multiusuario, incluso se llega a referenciar como un sistema operativo en si mismo... un cajon de sastre!

Segun el diccionario de IBM, su principal fabricante, un mainframe es ’un ordenador, generalmente de un centro computacional, con amplias capacidades y recursos a los que otros ordenadores pueden ser conectados para poder compartir facilidades (...).’ Algo asi como un procesador de la ostia con una gran base de datos y con capacidad de servir el resultado a otra maquina.

Otra definicion adecuada y muy flexible es la que sostiene que un mainframe es ’una plataforma de computacion de uso general en continua evolucion y que incorpora en su definicion arquitectural la funcionalidad esencial requerida por las aplicaciones a las que va dirigida’. Es una definicion mas basada en el software del cliente que en el hardware.

En mi humilde opinion, un mainframe es una gran maquina (o varias) de computacion, los sistemas operativos por los que corre, las arquitecturas de red que lo soportan, y las aplicaciones que ejecuta y que lo mantienen, y que posee una suprema capacidad para almacenar, procesar y compartir enormes volumenes de datos y de procesos de forma altamente rapida y extraordinariamente segura.

¨A que mola? Es un poco larga. ¨Como que no mola? Pues saca tu propia e igualmente valida conclusion despues de repasar la historia de los mainframes y las principales caracteristicas de los actuales.

3. Breve historia de los mainframes

La historia de los mainframes tiene 60 anos (12 lustros, maldita enye!).

Empieza con el nacimiento de maquinas enormes capaces por ejemplo de realizar multiplicaciones en pocos segundos (que ya es mas que muchas personas). P.ej., el SSEC de IBM de 1944, el ENIAC de 1942, BINAC de 1949...

Posteriormente, en la decada de los 50’s, nacen los ultimos ordenadores basados en tubos de vacio como los IBM 701, 650 y RAMAC 305, y los primeros basados en transistores como los IBM DPS 7070, 1401 y 7000. Consultad la web de IBM para un listado mas completo.

A partir de los 60’s los mainframes sufren un redisenyo radical. Aparece la serie IBM System/360, un sistema con tecnologia SLT (Solid Logic Tech.), economicamente mas asequible y con mejores prestaciones de almacenamiento, procesamiento y recuperacion de datos.

La decada de los 70’s trae la serie M de Hitachi y la serie IBM System/370 con

los recien nacidos circuitos integrados en placas de silicio. Se incorporan chips cada vez mas miniaturizados y con cableado de circuitos cada vez mas corto, sus sistemas de refrigeracion mejoran notablemente, empiezan a poder utilizar mas de un procesador en paralelo...
Los modelos IBM 30xx de la decada de los 80's parece que son mejoras de la serie System/370, con chips con mayores capacidades (p.ej., familia SAMOS) y con innovaciones en refrigeracion y mantenimiento del hardware.

Coincidiendo con la expansion y crecientes posibilidades del PC y de las work-stations, empiezan a escucharse malos augurios sobre los mainframes. Por entonces se les llego a dar menos de 10 anos de vida. Los fabricantes empiezan a fusionarse y a reestructurarse.

Pero con la llegada de internet, IBM introduce ya en los 90's la serie System/390 y la arquitectura ESCON z/os, dirigida al e-business de grandes empresas, y compuesta de circuitos ultradensos, comunicaciones por fibra optica, capacidades multiplicadas de supercomputacion y de optimizacion de los recursos (procesadores CMOS en vez de procesadores bipolares) que reducen el precio final.

Los 90's tambien traen la tecnologia de control en paralelo de los Hitachi MP5800.

El exponencial desarrollo de internet y la e-tecnologia ha traido la familia IBM eServer, con sus omnipotentes representantes z900, z800 y z990.

Con el anuncio de Hitachi de dejar de fabricar sus mayores modelos Trinium y Pilot se deja el monopolio en manos de IBM. El resto de mercado se lo reparten pocas companias mas, sobretodo asiaticas como Hitachi, Ahmdal, Fujitsu...

Las joyas actuales son los IBM z990. Salen a la venta en 2003. Introducen la puntera tecnologia MCM (MultiChip Module): un modulo mas pequeno que tu mano con 3 millones de transistores ultradensos aislados con tecnologia SOI capaces de procesar 9000 MIPS (millones de instrucciones por segundo). La virtualizacion que tienen les confiere la capacidad de soportar miles de servidores en un solo aparato. Posee exquisitos mecanismos dinamicos de redireccionamiento de recursos (y 256 GB de RAM). Y su seguridad y control de intrusiones y fallos es extrema. ¡¡Sus 32 procesadores creo que lo hacen mejor que mi Pentium II!!

Aunque tanta excelencia es muy cara: suelen costar entre 10.000 a 500.000 dolares USA.

4. Y quien puede permitirse unas maquinas tan caras?

Los mainframes son sistemas-maquinas-servidores que gestionan el procesamiento diario de cantidades desorbitadas de datos y de modo concurrente.
Por ello, son maquinas utilizadas basicamente por empresas que generan ese tipo de datos y que pueden pagar tantos millones (principalmente aerolineas y empresas de transporte, holdings bancarios, grandes almacenes...), centros superiores de investigacion privados y publicos, e instituciones publicas civiles y militares.
No obstante, en el terreno de la investigacion tambien tenemos a los monstruos capaces de realizar predicciones y simulaciones metereologicas, sismicas, militares, geologicas, fisicas... que no por grandes se llaman mainframes.

La mayor centralizacion del uso de mainframes para empresas privadas reside en las grandes capitales economicas y politicas (Madrid y Barcelona en el caso de Espana).

<Opinion> A pesar de crecer el volumen de transacciones y el mercado de negocio de estas empresas, que en ultima instancia se procesa en los mainframes, el maldito capitalismo subsubcontratista privado y amoral, se concentra en explotar y despedir al personal que lo mantiene funcionando.

5. Principales caracteristicas de un mainframe

Intentar encontrar puntos comunes a todos los mainframes actuales es tarea dificil. Basicamente son las siguientes, no necesariamente todas ni en el orden expuesto (ten en cuenta la definicion dinamica del mainframe segun el fin al que se destina):

- Aspecto: Las maquinas suelen ser del tamano de un armario o estanterias de biblioteca. Tienen gran ramificacion de conexiones y un sistema de refrigeracion fuera de lo normal (aire y/o agua). (En las referencias hay fotos).

- Elevadisima velocidad de ejecucion de instrucciones de maquina (cientos a miles de MIPS; MIPS = millon de instrucciones por segundo). Aqui se pueden englobar varias caracteristicas:
  1- Generalmente tienen varias CPU’s (hasta una trentena).
  2- Se comparten recursos por reasignacion dinamica de procesos al procesador disponible. P.ej., tecnologia IRD de IBM o PCT de Hitachi.
  3- Tecnologia VMS (Virtual Machine System): Permite soportar diferentes sistemas operativos en cada maquina virtual. El gran desarrollo de la virtualizacion amplian la memoria disponible y por tanto su poderio.
  4- Gran velocidad de I/O. La conectividad entre componentes (CPU o circuiteria de ejecucion, memoria o almacen de datos, y unidades perifericas) esta muy optimizada.
  5- Pueden incluso poseer un juego propio y mas amplio de instrucciones de maquina, de registros, de interrupciones... Un ejemplo es el High Level Assembly Language, que nada tiene que ver con el ensamblador para 80x86 que tanto me cuesta seguir.

- Elevadisimo numero de accesos a bases de datos sin cabida a la corrupcion de datos por contencion de procesos. Aqui juega un gran papel la tecnologia de la base de datos (DB2, IMS, Oracle...), el soporte (cinta, disco, cartucho, disco optico...), y el metodo de acceso (utilidades del fabricante). Hasta 7 Terabytes a disposicion de la empresa. Hasta varios millones de registros generados al dia a ser procesados en tiempo real (y en batch, ya no te cuento).

- Gran numero de usuarios en concurrencia (p.ej., miles de usuarios), gracias a tecnologia TSS (Time Share System).

- Uso continuado 365 dias al ano y 24 horas al dia. Permiten la reparacion de fallos sin dejar de dar servicio al usuario.

- El mainframe suele ser la maquina de ejecucion (jobs) y de almacen (bases de datos), y el resultado se envia a un terminal conectado (estaciones de tele-working).

- Soporta diferentes protocolos y arquitecturas de red y de comunicacion: SNA, TSO, emulacion T3270, TCP/IP, ftp’s, intranets y LAN’s... Creo que de todo.

- Los sistemas operativos soportados suelen ser especiales: z/OS, OS/390, MVS, VM, VSE, WindowsNT, UNIX e incluso Linux (almenos los IBM).

- Soportan aplicaciones generalmente programadas en COBOL, REXX, FORTAN, PLI, REXX, SAS, ASM, C, C++, Java, XML...

- Utilizan en inteligente sinergia las tecnologias mas punteras en seguridad contra errores internos y contra accesos no autorizados (p.ej. ACL o Access Control List, RACF o Resource Access Control Facility, DES o Data Encrypting Standard, firewalls y gateways, transacciones SSL o Secure Socket Layers, incluso pueden albergar utilidades criptograficas intrinsecas en la propia CPU...).

6. Conclusion

La vaguedad, amplitud y flexibilidad del concepto 'mainframe' hacen de el un sustantivo dificil de utilizar, a la vez que encaja en muchas definiciones.

En mi opinion, seria preferible hablar en terminos de sistema operativo o de la plataforma de comunicacion o de hardware que esta siendo utilizando, mas que de mainframe. O incluso de la empresa que lo utiliza.

Bueno, amiguetes, espero que os haya interesado.

7. Referencias

1.- http://www-1.ibm.com/ibm/history/exhibits/mainframe
Historia de los mainframes IBM.
2.- http://www-1.ibm.com/ibm/history/documents/pdf/glossary.pdf
Dictionary of IBM and Computing Terminology
3.- http://www.mainframes.com/whatis.htm
Lo que yo me pregunto: que es un mainframe.
4.- http://www.sdl.hitachi.co.jp/english
http://www.amdahl.com
Paginas de la principal competencia de IBM.
5.- http://www.thocp.net/hardware/mainframe.htm
Excelente web sobre la historia de la informatica. Visitadla!
6.- http://www.ibm.com
Miles de manuales tecnicos de productos IBM (serie RED books).

-[3x04]----------
-[interpolando]----------
-[by Fca0000]----------SET-29--

```
/*
Hola a todos, ninios y ninias. Bienvenidos a otra excitante leccion del
profesor Siesta.

Nuestro tema de hoy va a consistir en la imposibilidad de la interpolacion
numerica.Este es un tema para el que son necesarias unas bases matematicas
de las que se estudian cuando tiene 15 anios, que es aproximadamente cuando
se estudian polinomios en una variable.

Para los que lo han olvidado (o nunca lo han sabido), un polinomio de grado-n
es una funcion inyectiva, continua, derivable hasta n veces, con dominio R
(los numeros reales) y campo R o un subconjunto suyo, y que se expresa como
constantes multiplicativas de potencias de la variable.

Esto quiere decir mas o menos que:
-se definen como
```

```
 f(x)=a*x^n + b*x^(n-1) + c*x^(n-2) + ...+ t*x^2 + u*x + v
 Por supuesto, el simbolo '^' significar elevar a la potencia, o
 sea, multiplicarlo por si mismo tantas veces como indique
 el exponente)
-la variable  x  puede tomar cualquier valor Real. Esto
 incluye valores como 1, 5, 0, -17 , 24/5 , PI, raiz cuadrada de 2,
 raiz sexta de 17
-en cambio, valores no permitidos de x son: infinito, raiz
 cuadrada de -1 , matrices, conjuntos de numeros, valores
 de 2 dimensiones, y otros bichos raros
-cuando se toma una variable x y se le aplica la funcion f(x) se
 obtiene un valor  y. Normalmente se representa en un sistema
 de coordenadas, con una abcisa x y una ordenada y. Si, ese dibujo
 que tiene una raya vertical y otra horizontal y una curva continua.
-ser una funcion inyectiva significa que para un valor de x solo
 puede haber un valor de y. Por ejemplo, una circunferencia no
 puede provenir de un polinomio porque existen rayas verticales
 que cortan a la circunferencia en 2 puntos.
-es continua: si tomas un valor de x1 y otro cercano x2, sus
 respectivos valores y1 , y2 tambien estan cercanos. En otras
 palabras, no hay saltos al pintar la curva, ni huecos en medio.

Por ejemplo:
f(x)=x+9

f(x)=x^2 - 1

f(x) =x^400000 + x^ 399999

f(x) = (x-2) * (x+2)

f(x) = (x^4)/3 + 2*(x^3)/3 + 3*(x^2)/2 + 4*x + 5

f(x)=7

Pero NO son polinomios:
f(x) = 88*(x-5)/(x^4)

f(x) = sqr(x)  , donde sqr es la raiz cuadrada.

f(x) = x^x

f(x)= 2^x

f(x) = cos(x) , donde 'cos' es la funcion coseno

Lo que trato de mostrar aqui es que cualquier funcion cuyos
datos sean todos valores enteros (numeros sin decimales, vamos)
es posible expresarla como un polinomio
Por ejemplo:
si f(1)=2 , f(2)=4, f(3)=6, f(4)=8, f(5)=10, f(6)=12 , cuanto
crees que vale f(7) ?  La respuesta logica es 14, pues f(x)=2*x

Otro ejemplo:
f(1)=1, f(2)=4, f(3)=9, f(4)=16, f(5)=25, f(6)=36, f(7)=49,
cuanto crees que vale f(8) ? Esta vez la respuesta logica es 64, pues f(x)=x^2

Seguro que es mas dificil si uso polinomios con mayores
exponentes de x, y los valores de las constantes las uso
negativas y con fracciones (decimales). Pero no
se trata aqui de complicarnos la vida.
```

Una aplicacion inmediata es cuando se usa f(x)=g(f(x-1))
por ejemplo, para hallar el siguiente de una serie
de numeros. Esto es, en los que cada numero depende
solamente de su posicion en la lista.

Vamos alla: cual es el numero siguiente en esta serie:
2, 4, 6, 8, 10, 12, ? La respuesta logica es 14, pues
siguiente(x) = f(x) , siendo f(x)=x*2, siendo x su
posicion en la lista.

Y en esta serie:
0.0001 , 0.0004, 0.0009, 0.0016, 0.0025, 0.0036, 0.0049, ?
La respuesta es 0.0064 , y aqui f(x)=(x^2)/10000

Bien, pues ahora supongamos que tenemos unos numeros
de serie de un programa, o unos numeros de recarga de
tarjetas de moviles, o simplemente un test psicotecnico
de esos en los que piden cual es el numero que sigue a otros.

Concretando, la cuestion se reduce a:
partiendo de una serie de numeros (tan larga como se quiera)
nos piden averiguar el siguiente.

Bueno, pues la solucion es ’cual quieres que sea?’
En efecto, cualquier numero puede seguir la secuencia.

Por eso, si alguien me da una lista de 25 numeros
para recargar tarjetas de pre-pago de moviles yo soy
capaz de decir cual es el siguiente numero: 220 (pues
este es mi numero favorito), y soy capaz de demostrarlo.
Por supuesto que tambien puedo demostrar que es cualquier
otro numero, por eso no me sorprendo cuando me dicen el
siguiente numero y no coincide con el mio.

La demostracion mas convincente es dar la formula:

```
      N
   +-----        N
    \        ---------
     \        |     |
f(x)= \ { Y  *|     | (X - X)   /           }
      /    i  |     |       j  /  (X - X)
     /        |     |         /     i   j
    /         |     |
   +------     j=1
      i=1       j!=i
```

Que quiere decir este dibujo?
El primer simbolo se llama sumatorio. Quiere decir tomar
los valores desde 1 hasta N , hacer algo con cada uno de
ellos, y al final sumarlos todos.

El segundo simbolo se llama producto multiple (el nombre
correcto seria productorio). Quiere decir tomar los
valores desde 1 hasta N , pero solo si i es distinto de j,
hacer algo con ellos , y luego multiplicarlos todos.

Cada vez que aparece X_i o X_j quiere decir tomar el valor
de X en la posicion i-esima , es decir: la primera, la
segunda, la decimo-septima (es totalmente incorrecto

```
decir diecisieteava; eso significa otra cosa) o la
127-esima.

Similarmente, Y   es el valor conocido en el punto X
               i                                    i

Tomemos la serie 1,2,3,4 . El siguiente valor es 220.
No te lo crees?
Dicho de otro modo: f(1)=1, f(2)=2, f(3)=3, f(4)=4, f(5)=220
Para verlo, usamos este poligono interpolador, de acuerdo con
la formula que hemos dicho antes:


f(x)=   { 1 * [      (x-2)*(x-3)*(x-4)*(x-5)]/[      (1-2)*(1-3)*(1-4)*(1-5)] }
      + { 2 * [(x-1)*      (x-3)*(x-4)*(x-5)]/[(2-1)*      (2-3)*(2-4)*(2-5)] }
      + { 3 * [(x-1)*(x-2)*      (x-4)*(x-5)]/[(3-1)*(3-2)*      (3-4)*(3-5)] }
      + { 4 * [(x-1)*(x-2)*(x-3)*      (x-5)]/[(4-1)*(4-2)*(4-3)*      (4-5)] }
      + {220* [(x-1)*(x-2)*(x-3)*(x-5)      ]/[(5-1)*(5-2)*(5-3)*(4-5)      ] }

Efectivamente
f(1)=   { 1 * [      (1-2)*(1-3)*(1-4)*(1-5)]/[      (1-2)*(1-3)*(1-4)*(1-5)] }
      + { 2 * [(1-1)*      (1-3)*(1-4)*(1-5)]/[(2-1)*      (2-3)*(2-4)*(2-5)] }
      + { 3 * [(1-1)*(1-2)*      (1-4)*(1-5)]/[(3-1)*(3-2)*      (3-4)*(3-5)] }
      + { 4 * [(1-1)*(1-2)*(1-3)*      (1-5)]/[(4-1)*(4-2)*(4-3)*      (4-5)] }
      + {220* [(1-1)*(1-2)*(1-3)*(1-5)      ]/[(5-1)*(5-2)*(5-3)*(4-5)      ] }
calculando
f(1)=   { 1 * [          24                ]/[             24             ] }
      + { 2 * [ 0                          ]/[            -6              ] }
      + { 3 * [ 0                          ]/[             4              ] }
      + { 4 * [ 0                          ]/[            -6              ] }
      + {220* [ 0                          ]/[             24             ] }
O sea, f(1)=1*24/24 + 0 + 0 + 0 + 0 = 1

Igualmente f(2)=2 , f(3)=3, f(4)=4, y
f(5)=   { 1 * [ 0                          ]/[             24             ] }
      + { 2 * [ 0                          ]/[            -6              ] }
      + { 3 * [ 0                          ]/[             4              ] }
      + { 4 * [ 0                          ]/[            -6              ] }
      + {220* [  219 *  218 *  217 *  216  ]/[  219 *  218 *  217 *  216  ] }

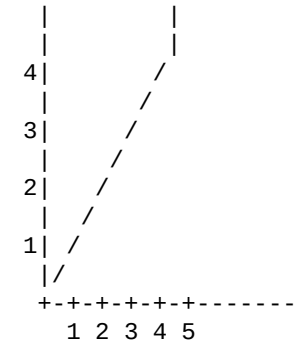

O sea, f(5)=0+0+0+0+ 220 = 220

Como queriamos demostrar.

Es posible que la gente a la que se lo cuentes no puedan entenderlo.
Bueno, las matematicas son sencillas cuando se tienen los conceptos claros.
Aqui no hay mas que 4 operaciones: sumas, restas, multiplicaciones
y divisiones.

Esta funcion es una curva algo asi:


220|          /
   |         |
   |         |
   |         |
   |         |
   |         |
   |        /
   |        |
   |        |
```




```
Otro ejemplo mas sencillo.
Si un crio de 2 anios bebe 0 litros de cerveza al mes, y un anciano
de 99 anios bebe 0 litros de cerveza, ?Cuantos bebe un chico
de 23 anios?
La solution, por supuesto, es 220
Simplemente porque la grafica de consumo de cerveza verifica que
f(1)=0
f(99)=0
f(20)=220
La funcion que cumple esto es:
f(x)= 0.0
     +0.0   * (        (x-23.0)*(x-99.0))/(           (1.0-23.0)* (1.0-99.0) )
     +220.0 * ((x-1.0)*         (x-99.0))/((23.0-1.0)*            (23.0-99.0))
     +0.0   * ((x-1.0)*(x-23.0)         )/((99.0-1.0)*(99.0-23.0)            )


Claro que esto dice que un chaval de 4 anios bebe 37.5 litros, pero
las matematicas no mienten... a menos que se les obligue.

Para finalizar doy un programa que dice la funcion que pasa por
unos puntos (x,y)
Cambia los parametros y sorprende a tus amigos !
Conviertete en el centro de atraccion de las fiestas !
Una diversion sana y barata !

*/
/* Este programa averigua una funcion que es capaz de interpolar
   unos valores. Sirve para demostrar que el siguiente numero
   en la secuencia
   10, 20, 30, 40, 50, ???
   debe ser 220
   Si quieres, modifica los valores y[1], y[2], y[3], ....
   Luego compila el programa y ejecutalo por primera vez.
   Escribira en pantalla una funcion       float f(x) ....
   Pega esas lineas otra vez en el codigo; compila y
   ejecuta de nuevo, y fijate en las ultimas lineas.
   Verificaran que esta funcion cumple los requisitos, y
   ademas mostrara otros valores
*/

#include <stdio.h>
#include <string.h>

#define N 6+1

/*  pega entre esta linea */
/*  ---------------------- */
float f(float x)
```

```
 {
 float f;

 f=0
+ 10.00 * (          (x-2.00)*(x-3.00)*(x-4.00)*(x-5.00)*(x-6.00) )/\
        ( (1.00-2.00)*(1.00-3.00)*(1.00-4.00)*(1.00-5.00)*(1.00-6.00) )
+ 20.00 * ( (x-1.00)*         (x-3.00)*(x-4.00)*(x-5.00)*(x-6.00) )/\
        ( (2.00-1.00)*         (2.00-3.00)*(2.00-4.00)*(2.00-5.00)*(2.00-6.00) )
+ 30.00 * ( (x-1.00)*(x-2.00)*         (x-4.00)*(x-5.00)*(x-6.00) )/\
        ( (3.00-1.00)*(3.00-2.00)*         (3.00-4.00)*(3.00-5.00)*(3.00-6.00) )
+ 40.00 * ( (x-1.00)*(x-2.00)*(x-3.00)*         (x-5.00)*(x-6.00) )/\
        ( (4.00-1.00)*(4.00-2.00)*(4.00-3.00)*         (4.00-5.00)*(4.00-6.00) )
+ 50.00 * ( (x-1.00)*(x-2.00)*(x-3.00)*(x-4.00)*         (x-6.00) )/\
        ( (5.00-1.00)*(5.00-2.00)*(5.00-3.00)*(5.00-4.00)*         (5.00-6.00) )
+ 220.00 * ( (x-1.00)*(x-2.00)*(x-3.00)*(x-4.00)*(x-5.00)          )/\
        ( (6.00-1.00)*(6.00-2.00)*(6.00-3.00)*(6.00-4.00)*(6.00-5.00)          )
;
 return f;
}
/*  ---------------------- */
/* y esta linea */

main(int argc, char *argv[])
{
int i=1, j=1, t;
float x[N], y[N];

for(i=0;i<N;i++)
        x[i]=0.0;
for(i=0;i<N;i++)
        y[i]=0.0;

x[1]=1.0;  y[1]=10.0;
x[2]=2.0;  y[2]=20.0;
x[3]=3.0;  y[3]=30.0;
x[4]=4.0;  y[4]=40.0;
x[5]=5.0;  y[5]=50.0;
x[6]=6.0;  y[6]=220.0;

printf("Copia estas lineas en el codigo ; compila, y ejecuta \n");
printf("para verificar que esta funcion interpola tus valores.\n");
printf("-----------------------\n");
printf("float f(float x) \n { \n float f; \n \n f=0\n");

for(i=1;i<N;i++)
        {
        printf("+ %3.2f * ", y[i]);
        printf("( ");
        for(j=1;j<N;j++)
                {
                if(i!=j)
                        printf("(x-%3.2f)",x[j]);
                else
                        printf("         ");
                if(j!=N-1 && i!=j && i+j!=N*N-2)
                        printf("*");
                }
        printf(" )");
        printf("/");
        printf("( ");
        for(j=1;j<N;j++)
                {
```

```
                if(i!=j)
                        printf("(%3.2f-%3.2f)",x[i], x[j]);
                else
                        printf("         ");
                if(j!=N-1 && i!=j && i+j!=N*N-2)
                        printf("*");
                }
        printf(" ) ");
        printf("\n");
        }
printf("; \n return f; \n}\n");
printf("-----------------------\n");
printf("Estos valores solo son correctos la segunda vez \n");
printf("que ejecutas el programa\n");

for(i=1;i<N;i++)
        {
        printf(" f(%3.2f)=%3.2f \n", x[i], f(x[i]));
        }
printf("Otros valores:\n");
printf(" f(0)  =%8.2f \n",   f(0));
printf(" f(%3.2f)  =%8.2f \n", (float)(N),  f((float)(N)));
printf(" f(1.5)=%8.2f \n",   f(1.5));
printf(" f(100.0)=%8.2f \n", f(100.0));
printf(" f(999.9)=%8.2f \n", f(999));

return 1;
```

EOF

```
-[ 0x04 ]----------------------------------------------------------------
-[ Logs en Linux ]--------------------------------------------------------
-[ by karmax ]-----------------------------------------------------SET-29--
```

LOGs en LINUX

Quienes me conocen saben que considero fundamental revisar los logs del sistema, ya que considero una de las mejores maneras de prevenir un posible ataque o tambien un error en el sistema. Si bien hablare de logs de linux, este concepto se aplica a cualquier sistema operativo.

Como sabran *nix intenta que todo sea un archivo, desde un dispositivo como el HD hasta un archivo de configuracion y los logs no son la excepcion.

Basicamente el logging en linux esta basado en syslogd:

SYSLOGD

"syslogd": es quien proporciona el servicio al sistema para poder logear segun este especificado en su archivo de configuracion.

Podemos acceder a su archivo de configuracion en /etc/syslog.conf en donde podremos observar que a grandes rasgos se encarga de indicar que y donde se envian los logs. Esto no quita que existan programas que se encargan de administrar sus propios logs (por ejemplo Apache).

```
Observamos con atencion unas lineas del archivo syslog.conf
kern.*                          /var/log/kern.log
mail.*                          /var/log/mail.log
(A)                                     (B)
```

NOTA: A Y B fueron agregados por mi para organizar el siguiente tema.

Esta compuesto principalmente de dos partes: -Que logear (A)
-Donde enviarlo (B)

Que logear

Se indica el servicio que envia el mensaje y la prioridad del mensaje, separados por un punto "."

Los servicios: auth, auth-priv, cron, daemon, kern, lpr, mail, mark, news, syslog, user, uucp, local0 a 7

Las Prioridades: debug, info, notice, warn, err, crit, alert, emerg.

Parametros especiales:

* (asterisco)

Cumple la misma funcion que es utilizada en bash, indica "todos". Pero en el archivo de configuracion indicara TODOS, segun la posicion en la que se encuentre.

Veamos un ejemplo:

```
kern.*                          /var/log/kern.log
```

En este caso el servicio es el kern (kernel) y el nivel de alerta que deseamos es * (todos), de esta manera estariamos enviando TODOS los mensajes del servicio kern a /var/log/kern.log

, (coma)

Se utiliza para indicarle a varios servicios una misma prioridad.

Por ejemplo:

```
kern,mail.warning                       /var/log/kern_mail.log
```

Se enviaran a /var/log/kern_mail.log todos los mensajes que produzcan los servicios kern y mail, de prioridad warning (4) e inferiores, err (3), crit (2), alert (1), emerg (0).

= (igual)
Es utilizado para dar una prioridad especifica a uno o mas servicios.
Por ejemplo:

```
kern,mail.=warning                          /var/log/kern_mail_warn.log
*.alert                                     /var/log/alerts.log
```

En el primer caso se logeara todos los mensajes de prioridad warning (4), que produzcan los servicios kern y mail. En el segundo todos los mensajes de prioridad alert (1) que produzca cualquier servicio seran logeadas en el archivo /var/log/alerts.log

! (exclamacion)
Se utiliza para exceptuar la prioridad mas las inferiores, tambien se puede utilizar con el signo = para exceptuar solo esa prioridad.
Por ejemplo:

```
kern.notice;kern.!crit                      /var/log/kern_notice.log
mail.*;mail.!=notice                        /var/log/mail.log
```

Aqui el servicio kern logeara los mensajes de prioridades notice (5), warning (4) y err (3). Esto se debe a que con el campo !crit estamos exceptuando todas las prioridades crit (2) e inferiores: alert (1), emerg (0).
En la segunda linea podemos observar que se utiliza != para exceptuar los mensajes de prioridad notice (5). Es decir que se logearan todos los mensajes que produzca el servicio mail, menos aquellos que sean de prioridad notice (5).

Donde enviarlo

En este campo se especifica donde se enviaran los logs: si seran guardados, impresos o si directamente se guardaran en otro servidor, etc. Generalmente son enviados a archivos de texto plano, los cuales deben estar correctamente especificados y con su path completo. Tenemos una opcion bastante util, que nos permite evitar una sincronizacion constante del archivo en el que se esta logeando, de esa manera se puede conseguir un mayor rendimiento en velocidad. Para utilizarla simplemente debemos anteponer el signo - antes del path. Si bien se pueden perder datos si hay un error en el sistema al momento de escribir el log, considero que se puede utilizar sin miedo, dependiendo del sistema y de lo que estemos logeando.
Hasta ahora vimos como enviar los logs a archivos de texto ubicados en nuestro sistema, veamos otras opciones:

DEVICES (/dev/)

Podemos enviar los logs a dispositivos (Devices) de nuestro sistema directamente a una terminal determinada.
Por ejemplo:

```
kern.emerg                                  /dev/console
mail.emerg                                  /dev/console
```

Seran enviados a la consola (por pantalla) todos los mensajes de prioridad emerg (0) de los servicios kern y mail.

PIPES (|)

Podemos enviar los logs a un archivo "pipe", en este tipo de archivo por decirlo de una manera entendible, se almacenan para ser leidos mas tarde.
Por ejemplo:

```
Primero creamos el archivo "pipe":
root@utopia:~# mkfifo tubo
root@utopia:~# chmod 0 tubo
root@utopia:~# ls -l tubo
p---------    1 root   root          0 Oct 20 19:30 tubo
```

Recordemos que syslogd debe estar configurado para enviarlo ahi:

```
mail.emerg                                          |/var/log/tubo
```

Como notaran antes del path esta el caracter "pipe" | de esa manera le indicamos que se trata de un archivo pipe a syslogd. Una vez realizado esto solo basta leerlos:

```
root@utopia:~# cat tubo
```

Se puede jugar mucho con esta funcion, despues veremos un ejemplo MUY UTIL de esa funcion.

OTRO HOST

Siempre que sea "necesario" y se pueda, recomiendo utilizar otro host para logear, ya que de esa manera prevenimos que sean borrados o modificados, logrando asi mayor confianza en nuestros logs. Suponiendo que tenemos un servidor (web, mail, ftp) debemos reconocer que existe la posibilidad de que alguien logre vulnerar nuestro sistema, lo grave seria no darnos cuenta.
Al tener un host aislado (solo accesible desde el servidor) y tomando las medidas de seguridad necesarias (fw, attributos de archivos, usuarios, contrase~as), podemos tener una confianza bastante alta sobre lo que dicen nuestros logs. Si los logs se realizaran en el mismo servidor y alguien lograra vulnerar el sistema, quizas ni siquiera lo notemos ya que podria ocultar procesos, usuarios, modificar logs y nosotros ni siquiera nos enterariamos.
Para enviar a otro host simplemente basta con una @, veamos...
Por ejemplo:

```
*.info                                              @host.seguro
```

Aqui se enviarian todos los mensajes de prioridad info (6) e inferiores, de cualquier servicio, a host.seguro
Si bien syslogd nos permite de manera MUY simple logear en otro host, tenemos que asegurarnos que este configurado correctamente para poder realizarlo. En el host.seguro debemos utilizar la opcion -r de syslogd para que acepte conexiones remotas (por default deshabilitado).
Al utilizar un nombre de host es conveniente tenerlo definido en el sistema (/etc/hosts), aunque syslogd intentara 10 veces acceder a ese host antes de ANUNCIAR que no puede logear en el sistema remoto. (Las conexiones se realizan a traves del puerto 514/udp y deben asegurarse de reiniciar el servicio syslogd en el host.seguro con la opcion -r; tambien recuerden que los logs se envian en texto plano)
Lo que parece un GRAN aumento de confianza y seguridad a nuestros logs puede representar un GRAVE fallo de seguridad si no se realiza correctamente. Veamos

porque:

- LAN

HUB

Nuestra LAN esta conectada a traves de un HUB, por lo que cualquier host de la red podria capturar los paquetes que sean enviados de un host al otro (sniff) y obtener informacion muy valiosa, nuestros LOGS.

SWITCH

Quizas consideremos que al utilizar un switch evitamos que cualquier host de la red pueda capturar paquetes, pero no es asi. Hoy en dia existen herramientas que permiten de manera muy simple "enga~ar" las tablas ARP, para poder continuar capturando paquetes. Si quieren mas informacion respecto a este tema, esto se denomina "ARP POISONING" o "ARP SPOOFING" y ettercap es una herramienta que permite hacerlo de manera MUY sencilla. En proximos boletines se tratara el tema.

- DIRECTO

Podriamos establecer una UNICA conexion FISICA entre el servidor y el host.seguro encargado de los logs, de esa manera se aumenta la seguridad. Por ejemplo se podria utilizar una tarjeta para conectar los dos hosts y asi evitar el trafico por el resto de la red. De esta manera estamos aislando al host.seguro del resto de la red y haciendolo accesible solo por el servidor.

- SEGURIDAD

CONEXION SEGURA

Lo que yo recomiendo es utilizar una conexion segura (SSH) al transmitir los logs a otro host. La comunicacion entre los hosts es cifrada por lo que si un atacante pudiese obtener los paquetes que se transmiten no lo servirian de mucho ya que la conexion es cifrada. Veamos como hacerlo:
Para poder hacer esto necesitamos trabajar con archivos "pipe" (|), para crear este arhivo simplemente:

```
root@utopia:~# mkfifo tuberia
root@utopia:~# chmod 0 tuberia
root@utopia:~# ls -l tuberia
p---------    1 root   root          0 Oct 20 19:30 tuberia
```

Ahi ya tenemos creada nuestra tuberia ahora como habiamos visto antes, para indicarle a syslogd que queremos enviarlo a un arhivo "pipe" simplemente antes del path destino agregamos |

La linea se veria asi:

```
kern.*                                  |/var/logs/tuberia
```

Una vez realizado esto los logs del servicio kern de cualquier prioridad seran enviados al archivo pipe "tuberia", ahora debemos enviarlo por ssh al otro host. Lo ideal seria tener esta conexion y todas las referentes al log en scripts de manera tal que se realizen constantemente desde el inicio del sistema. Es cierto que necesitara mas atenciones pero creo que si la seguridad importa, es algo que vale la pena.
Para mas info en "man syslogd" hay una seccion que hace referencia a este

tema, explicando otros casos.

FIREWALL

Es fundamental tener configurado nuestro firewall de manera que no entorpezca las conexiones, pero que ASEGURE el aislamiento del host en una conexion DIRECTA asi tambien COMO TODA LA RED. Ya se hablo en boletines anteriores sobre Firewall veanlos (Boletines 2 y 3) igualmente NUNCA dejaran de ser tema de conversacion ;)

- IMPRESORA (/dev/lp1)

Si ni siquiera el metodo anterior les parece aun seguro nos queda imprimir nuestros logs a medida que se producen, a muchos les parecera un poco mucho y otros perfectamente normal. Podemos configurar syslogd de manera ciertos mensajes sean enviados directamente al dispositivo de impresion.
Esto es verdaderamente muy simple:

```
*.warn                                          /dev/lp1
```

Todos los mensajes de cualquier servicio de prioridad warn seran enviados a la impresora (/dev/lp1).

- USUARIOS CONECTADOS

Podemos enviar mensajes a usuarios logeados en el sistema.
Por ejemplo:

```
*.crit                                          root, admin
*.=emerg                                        *
```

Todos los mensajes de prioridad crit (2) e inferiores de cualquier servicio seran enviados a los usuarios root y admin. Podemos indicar tantos usuarios como deseemos. En el segundo caso vemos como root y admin fueron reemplazados por un asteriso (*), de esta manera se enviaran todos los mensajes de prioridad emerg (0) de cualquier servicio a todos los usuarios conectados en el sistema.

OTROS CONCEPTOS

Ademas de lo que hablamos hasta ahora hay otros conceptos que debemos conocer:

- logger

Este comando nos permite interactuar con syslog desde la shell, quizas en principio no les parezca muy util, pero a medida que vayan utilizandolo y agregandolo en sus scripts, veran lo util que resulta.
Veamos algunos ejemplos:

```
karmax@utopia:~$ ls && logger -p user.info -t comando "ls se ejecuto
correctamente"
root@utopia:~# tail -1 /var/log/messages
Oct 18 06:58:49 utopia comando: ls se ejecuto
```

Veamos cada opcion que utilizamos:

-p user.info

```
Como ya vimos en syslog.conf aqui debemos especificar servicio.prioridad

-t comando
Es el "tag" que le damos al mensaje.

"ls se ejecuto"
Es el mensaje en si, lo que se logea.


karmax@utopia:~$ logger -p user.info -f /home/karmax/pepe -t pepe
root@utopia:~# tail -1 /var/log/messages
Oct 18 07:12:18 utopia pepe: rocknroll

-f /home/karmax/pepe
Aqui simplemente enviamos el contenido de este archivo como un mensaje de
user.info

   Recuerden que con logger estan produciendo un mensaje de un servicio y
prioridad determinada, el cual syslog detectara y lo enviara segun hayamos
indicado en su archivo de configuracion.
   Como veran ambos ejemplos no son muy utiles a lo que comodidad y seguridad
refiere, pero se los dejo para que hagan los suyos. Con logger se puede hacer
mucho mas, para mas info: "man logger"
   Si lo suyo es programar en C, les recomiendo "man syslog"


- klogd

   Se encarga de logear los mensajes del kernel, al estar separado de syslogd,
ofrece una clara separacion de servicios. Hay dos fuentes principales de
informacion referentes al log del kernel. Primero intenta acceder a /proc/kmsg
si no puede (no esta montado el fs /proc) utiliza la llamada a sistema
"sys_syslog" para obtener los mensajes del kernel.
   Si los mensajes del kernel son enviados a syslogd, klogd cuenta con la
capacidad de darle una determinada prioridad al mensaje. Por default todo lo
que sea inferior a 7 se vera en pantalla. Supongamos q nosotros solo queremos
ver los mensajes (3, 2, 1), entonces deberiamos utilizar la opcion -c del
klogd:
klogd -c 4
Una lista de a que corresponde cada cada nivel de mensaje:
#define KERN_EMERG      "<0>"   /* system is unusable                   */
#define KERN_ALERT      "<1>"   /* action must be taken immediately     */
#define KERN_CRIT       "<2>"   /* critical conditions                  */
#define KERN_ERR        "<3>"   /* error conditions                     */
#define KERN_WARNING    "<4>"   /* warning conditions                   */
#define KERN_NOTICE     "<5>"   /* normal but significant condition     */
#define KERN_INFO       "<6>"   /* informational                        */
#define KERN_DEBUG      "<7>"   /* debug-level messages                 */
Esta lista se encuentra en kernel.h para poder verla simplemente hay que hacer
lo siguiente:
karmax@utopia:~$ cat /usr/include/linux/kernel.h |egrep "\"<"
El klogd es mucho mas que esto y si les interesa les recomiendo "man klogd"


- Seguridad general

   Si bien no podemos tener una confianza ABSOLUTA en lo que dicen nuestros logs
(ya que pudieron haber sido modificados/evitados) tenemos que intentar llevar
esta confianza al maximo, y para esto es necesario asegurarlos.
   Como administradores de sistema no queremos que nadie vea nuestros logs y
mucho menos que lo pueda modificar o borrar. Imaginen que alguien tenga acceso
a sus logs, bastaria analizar sin despertar ninguna alerta buscando el momento
```

oportuno para realizar el ataque.. no le resten importancia a los LOGS!
   Como primera medida aseguremonos que los archivos de log solo sean accesibles para root y su grupo.

Para cambiar de owner y grupo:
chown owner grupo archivo
(donde archivo puede ser un directorio)
Por ejemplo:

```
root@utopia:~# chown root root /var/log/user.log
root@utopia:~# chmod 600 /var/log/user.log
Ahora comprobamos...
root@utopia:~# ls -l /var/log/user.log
-rw------    1 root     root      2188 Oct 15 23:41 /var/log/user.log
```

Realizenlo con los archivos que crean necesarios

   Tambien podemos utilizar el comando chattr para cambiar los atributos de archivo, nos convendria utilizar la opcion +a (de esa manera solo se puede agregar contenido al archivo). Si tenemos algun programa como logrotate deberiamos incluir en el script del programa un -a para poder dejarlo en 0 y luego un +a para volver a indicarle el attributo.

- Archivos principales

   En syslog vienen definidos por default algunos logs, entre ellos se encuentran:
NOTA: Recuerden que el archivo syslog.conf puede ser configurado para logear lo que quieran, aqui comento lo "general", asi tambien como cuando indique la linea de syslog.conf correpondiente a ese log, sera a modo de ejemplo.

/var/log/auth.log

   Es en donde se guardan logs referidos principalmente con la "seguridad" del sistema.

/var/log/messages

```
*.=info;*.=notice;*.=warning    /var/log/messages
```

   En este archivo estan la mayoria de los mensajes, a nivel informativo asi tambien como nuestro el log que realiza el firewall (NetFilter)

/var/log/syslog

   En general se logea todo lo relacionado con accesos (o intentos) a los servicios del sistema (telnet, rpc). Asi tambien como algunos avisos referentes a los modulos.

/var/log/debug

```
*.=debug                        /var/log/debug
```

   Es donde se registran los mensajes de prioridad debug (6), si lo observan veran que la mayoria son producidos por el kernel.

/var/log/daemon.log

```
daemon.*                        /var/log/daemon.log
```

   Contiene un log de los Daemons (Servicios) que se cargan en el sistema asi tambien como informacion referente a avisos en los modulos.

```
/var/log/kern.log

kern.*                          /var/log/kern.log
```

   Los mensajes producidos por klogd en su mayoria son almacenados aqui.

```
/var/log/user.log

user.*                          /var/log/user.log
```

   En su mayoria son mensajes enviados a usuarios en el sistema, como por ejemplo un aviso de "shutdown".

```
/var/log/mail.log

mail.info                       /var/log/mail.info
mail.warning                    /var/log/mail.warning
mail.err                        /var/log/mail.err
```

   Informacion de cada mensaje que entra y sale de nuestro servidor, asi tambien como detalles sobre el mail en cuestion.

```
/var/log/wtmp
```

   Este archivo contiene la informacion relacionada a los logins y logouts relacionados al sistema (de donde, horario, fecha, usuario..) para poder verlo tenemos que utilizar el comando "last". Si quieren mas informacion al respecto "man wmtp" y "man last".
Por ejemplo:

```
root@utopia:~# last -2
karmax   tty7                        Sat Oct 18 07:57   still logged in
karmax   tty8                        Sat Oct 18 07:56   still logged in
```

   Vemos cuales son los 3 ultimos logins o logouts
   Relacionado con este, tambien tenemos el archivo "utmp", el cual nos brinda informacion acerca de los usuarios logeados en el sistema al momento de solicitar la informacion. Para acceder a esta informacion podemos utilizar el comando "who" (tambien se puede utilizar el last). Mas informacion "man who".

```
/var/log/lastlog
```

   Este archivo contiene informacion relativa a la fecha y hora del ultimo login de cada usuario, para acceder a esta informacion podemos hacerlo con el comando "lastlog". Mas info "man lastlog". Por ejemplo:

```
root@utopia:~# lastlog -u karmax
Username         Port     From             Latest
karmax           tty8                      Sat Oct 18 07:56 -0300 2003
```

   Para indicar de que usuario queremos saber anteponemos -u al nombre de usuario.

/var/log/faillog

   Tambien tenemos el archivo "faillog" el cual indica los intentos fallidos de cada usuario al querer logearse en el sistema, para acceder tenemos el comando "faillog". Mas info "man faillog". Por ejemplo:

```
root@utopia:~# faillog -u karmax
Username   Failures  Maximum  Latest
karmax            0        2
```

   En la columna "Maximum" se indica la cantidad de intentos erroneos que se permiten antes de deshabilitar la cuenta, para modificar este valor "faillog -u karmax -m valor" donde valor es el numero maximo. Recuerden que para realizar esto necesitan tener permisos de escritura sobre /var/log/faillog dejenle acceso de escritura a este archivo solo a root (por default).

sulog

   Este archivo contiene la informacion referente al comando "su" (man su), utilizado para cambiar el user ID. Esta informacion es bastante importante ya que logeariamos usuarios que utilizaron el comando su para, por ejemplo, obtener permisos de root. Para poder logearlo primero debemos comprobar el archivo /etc/login.defs

```
root@utopia:~# cat /etc/login.defs |fgrep "SYSLOG_SU_ENAB" -A 6
SYSLOG_SU_ENAB          yes
SYSLOG_SG_ENAB          yes

#
# If defined, all su activity is logged to this file.
#
SULOG_FILE     /var/log/sulog
```

   Veamos los campos importantes:

```
SYSLOG_SU_ENAB          yes
Ahi le indicamos que queremos que syslog logee su

SULOG_FILE     /var/log/sulog
En donde queremos que se logee
```

   Tengan en cuenta que el archivo login.defs es MUY importante para la seguridad y configuracion de nuestro sistema, en otra ocasion hablare de este y otros archivos de configuracion del sistema.

- Bash

   Si bien hay muchisimas shells en linux, hablare de Bash ya que es la mas conocida. En bash tenemos lo que se denomina Historial, el cual segun como lo configuremos, podra guardar los comandos que hayamos ingresado. Para configurar esto, lo podemos hacer desde el archivo de "profile" tanto el global (para todos) "/etc/profile" o para cada usuario "~user/.bash_profile".
   En mi opinion podriamos configurarlo de manera tal que los datos no sean modificables (con el chattr visto anteriormente) y ademas asegurarnos de que cada usuario SOLO tenga acceso a su directorio, para evitar que se anden leyendo los historiales entre ellos.
   Recordemos que puede ser un error MUY GRAVE de seguridad permitir que otros usuarios tengan acceso a esto. Ya que puede contener informacion muy importante que hayamos tipeado, nombres de archivo, acciones que realizamos, etc. Tengamos

cuidado con esto.

Bueno, espero que les sea de alguna utilidad este texto, prometo ampliar este tema en otra ocasion, pero por hoy lo dejamos asi.
Sigan disfrutando del boletin.

KarMax

EOF

```
-[ 0x05 ]----------------------------------------------------------------
-[ El SO de mi vida... (PARTE II) ]----------------------------------------
-[ by Kstor ]-------------------------------------------------------SET-29--
```

LLegamos a la segunda parte del El SO de mi vida..., esta serie de art. tiene como objetivo (para mi) el conocimiento de los distinto SO que hay.
Ya se que existen otros (muchos otros!), pero estos me gustan y quiero compartirlo con ustedes.
Las personas que esperen datos tecnicos no se enojen :), y leanlo igual.

Con esto termina el tema y para la SET proxima tendre algo nuevo para ustedes, ... espero.... :p

****
//Indice\\
  ******

##

Unix el sistema que a todos nos gusta, menos a mi viejo jeje, pero bueno...

#1

El origen de este SO data del a~o 1962 cuando el MIT y CTSS investigan en areas de tiempo compartido y proteccion.
En 1995 la division de investigacion de AT&T (Bell Labs), General Electric y el MIT trabajan en un proyecto de Multic, se trataba de un procesador con gran potencia de calculo y almacenamiento de muchos usuarios. De donde se obtuvieron muchos resultados, tanto como las shells, capacidad de multiprocesos y el arbol de archivos.
En 1969 fue abandonado y asi surgio de hay Ken Thompson que desarrollo un sistema de archivos propio, lo que queria el era derrotar al imperio Klimgon de Star Trek y monto una simulacion de la galaxia.
Entonces aparecio Dennis Ritchie y con una computadora PDP-7 monto el sistema para poder jugar.
Asi se dice que empezo la historia de Unix.

En el a~o en 1971 Richie y Kernigham crean el lenguaje C y lo utilizan en una PDP-11.
Dos años despues se juntan otra ves Thompson y Richie reescriben su sistema en C y forman el monstruo UNIX.
Demostro que era un sistema muy bueno y fue adoptado por 25 instalaciones en Bell Labs.
Crecio y crecio, en 1978 ya habia 100000 sistema UNIX en todo el mundo...

Unix llego a universidades y asi fue el caso de Berkeley en California que

modifico el sistema e incorporo el sistema de memoria virtual paginada.
En 1978 lanzaron el Unix 3BSD. luego con la subvencion de ARPAnet surge 4BSD.
Un año despues la version 4.2BSD donde adopta SUN Microsystems.

Y 1984 se marca un punto muy inportante ya que SUN desarrolla los conceptos de
RPC (remote proceduce call) y NFS (network file system) y los incorpora a
SunOS.

En general paso esto:

1969 - Se desarrolla UNIX por Thompson y Richie
1970 - Version dos usuarios sobre plataforma PDP-11
1971 - Version multiusuario en plataformas PDP-11/40/45/70
1973 - Se reencribe en lenguaje C por parte de Richie y Kernigham.
       Originalmente estaba programado en ensamblador. Ahora se podia
       transportar a otras maquinas.
1974 - Empieza a venderse la fuente (USD 25000)
1975 - Version 6 de Unix y se emplemento en Universidades para su estudio.
1983 - Aparece Unix System V con soporte para mensajes y memoria compartida
1989 - Unix System V, R4 que contenia soporte para RFS, Redes y tiempo real
1993 - Novell compra Unix a AT&T
1994 - Novell le da el nombre de Unix a X/open
1995 - Santa Cruz Operations compra UnixWare a Novell y se asocia con HP
       anunciando
       que lanzaran una version de Unix de 64-bit.
1996 - Hay mas de 3 millones de computadores con Unix a nivel mundial.
2003 - Muchas mas... :-)

#1.2 ** Caracteristicas de Unix en general:
     ****

* Bastantes herramientas de Software (compiladores, editores, uilitarios, etc)

* Reutilizacion de sofware a traves de comandos simples en aplicaciones
  complejas.

* Portable: puede correr en cualqier tipo de maquina, desde una NoteBooks a
  una super computadora. Es el unico SO que permite hacerlo

* Flexible: se adapta a muchas aplicaciones diferentes

* Potente: dispone de muchos comandos y servicios ya incorporados

* Multiusuario: permite ser usado por muchos usuarios al mismo tiempo

* Multitarea: permite la ejecucion de diferentes tareas al mismo tiempo

* Elegante: sus comandos son breves, coherentes, especificos para cada tarea y
            muy eficientes.

* Orientado a redes desde el comeinzo

* Dispone de un standar (POSIX) que debe cumplir todos los sitemas que
  pretenden ser Unix, lo que asegura un evolucion predecible. Tiene incluido
  en su arquitectura el protocolo TCP/IP

* El nucleo es muy compacto en relacion con otros sistemas operativos de
  tiempocompartido. Introduce la idea de reducir el Kernel (nucleo del sistema)
  y ceder ciertas funciones a funciones externas al nucleo llamados Demonios

* Los usuarios se encuentran divididos en dos grupos principales, el root
  (el DIOS de Unix) y los usuarios normales controlados por el SO segun las
  directivas del root.

```
* El sistema de archivos esta basado en volumenes que se pueden desmontar y
  montar un sistema fisico puede dividirse en en uno o mas modulos

#2 * Clases de Unix:
   ****************
-----------------------------------------------
#2.1 Linux:

Creado por Linus Torvalds, un programador Finlandes de 21 a~os que queria crear
un sistema operativo para su uso personal.
Linus usaba una peque~a version de Unix llamada Minix.
A mediados de 1991 el queria hacer un kernel para un nuevo sistema operativo,
parecido a Minix, pero mejor.
Con el tiempo fue desarrollando su SO y se convirtio en un proyecto mas serio.
El 5 de Octubre publico en en un grupo de noticias en Internet un articulo que
hablaba sobre un nuevo SO llamado Linux (v. 0.02) que pronto lanzaria para que
cualquier persona lo descargara gratis e invito a programadores para que lo
corrigieran y le reportaran errores.

En 1992 ya Linux contaba con 100 usuarios que corregian errores que mandaban a
Torvalds por Internet.
Ese mes salio la version 0.12 de Linux que incluia partes de codigo de otros
programadores y la primera que se desempeñaba mejor que Minix.

El numero de usuarios fue aumentando y no era estraño ya que en esa epoca los
estudiantes utilizaban Unix ya que les parecia estable y potente, pero costaba
4000 dolares la version.
En cambio Linux era un sistema Unix gratuito y funcionaba en procesadores Intel.
Cada ves aumentaron mas los programadores que ayudaban al desarrollo del sistema.
Linus entonces empezo a distribuir Linux bajo una licencia GPL, que permitia a
cualquier persona bajar el SO, modificar e inluso venderlo sin pagar un peso.
La unica condicion era que cualquier modificacion o campa~ia que se realizara
con Linux deberia ser publica.
Se calcula que gracias a esto Linux tenia 2000 usuarios a fines del año.

1993 habia 20000 usuarios y mas de 100 programadores ayudando al desarrollo.
En el mundo Linux aparecen actualizacion cada dos o tres dias. No cada 2 o 3
a~os como los programas comerciales.

En Marzo del 94 se lanzo la version completa de Linux 1.0, ofrecio soporte para
redes y muchisimas utlidades.
Se fundo Red Hat Software uno de los principales distribuidores de Linux.
Todos sabemos que Linux se puede bajar gratis de internet pero estas empresas
elaboran sus propias versiones y las venden en CD, junto a aplicaciones,
manuales y soporte tecnico. Su valor van desde 10 a 70 dolares segun la empresa.
En estos tiempos existian 100000 usuarios de LInux.

En el 95 ya habia 500000 (crecia rapido el nene...)

En el 96 se lanza Linux 2.0 que incluia soporte para dos procesadores. Y ademas
de soportar los procesadores x86 de Intel tambien soportaba los Alpha.
Los usuarios eran 1.5 millones (crecia mas rapido que mi Tamagochi)

En 1997 Linus fue a vivir a Santa Clara (California, USA) ya que fue contratado
por una compania que producia chips que no tenia nada que ver con Linux.
Pero igual siguio con el proyecto. Ese año lanzo la version 2.1
Usuarios: 3.5 millones...

En 1998 las empresas anunciaron que sacarian porgramas para Linux esto ayudo
mucho a su desarrollo. Ej. NetsCape, Computer Asscociates (CA), Oracle, etc.
En Diciembre Corel saco una version de su procesador de textos WordPerfect 8
para Linux.
Usuarios: 7.5 millones y mas de 100000 programadores en el mundo
```

En 1999 se lanzo la version 2.2 que ofrecia soporte para procesadores Sparc,
Motorola 68000, PowerPC y Mips.
Soportaba hasta 8 procesadores.
Corel anuncio que sacaria una version de Corel Linux
Linus dijo que a finales de ese año lanzaria la version 2.4 del Kernel de Linux
que mejoraba la funcionalidad a multiprocesamiento simetrico.

Hasta estos dias Linux es uno de los sistemas mas usados y se calcula que cuenta
con mas de 10 millones de usuarios.

Caracteristicas:
****

- Multitarea
- Multiplataforma
- Codigo fuente disponible
- Librerias compartidas de carga dinamica
- Control de tareas POXIS
- Consolas virtuales multiples
- Multiusuario
- Soporte para varios sistemas de archivos
- Soporte protoclo TCP/IP
- Carga de ejecutables por demanda
- entre otros...

** Algunas distribuciones Linux: (No todas)
****

* RedHat:

- Buena calidad
- Facil de instalar, incluye programas de configuracion que simplifican la tarea
- Buena documentacion, pero esta en Ingles :(
- Buen seguimiento de fallos y las correcciones se pueden bajar de Internet
+ www.redhat.com

* Debian:

- Muy buena calidad
- No esta sometida a preciones comerciales. Antes de su lanzamiento cuidan mucho
  su calidad
- Definida como la distribucion para los mas avnzados. Es un poco dificil de
  instalar para el usuario novato o sin conocimiento de informatica.
- Buena documentacion, en Ingles
- Ls arreglos de errores pueden bajarse de Internet
+ www.debian.org

* Caldera:

- Buena calidad
- Facil de instalar
- Buena documentacion, en ingles.
- Las correcciones de fallos pueden bajarse de la web.
+ www.caldera.com

* SlackWare:

- Fue una de las primeras y por lo tanto la que mas se utlizo. Durante un
  tiempo no se ha actualizado y perdio algunos seguidores
- Un poco dificil de configurar
- Documentacion en Ingles
+ www.slackware.com

```
* Mandrake:

- Buena calidad
- Facil de instalar
- Optimizada para procesadores Pentium y superiores
+ www.linux-mandrake.com/es/

* Conectiva:

- Buena calidad
- Facil de instalar
- Gano mucho respeto en el mercado Sudamericano y Espa~ol (esta en espa~ol):)
+ www.conectiva.com

* Hispafuente: (otra en espa~ol)

- Buena calidad. Esta basada en RedHat Deluxe y es 100 % compatible con ella.
- Facil de instalar
- Ha entrado con mucha fuerza en el mercado Espa~ol
- Toda la documentacion de Linux se encuentra en castellano
+ www.esware.com

/* Comentario de KSTOR */

Existen minidistribuciones de Linux que entran en un Diskette,
pueden servir para disco de rescate o para probar.
Algunas son:

Trinux - http://www.trinux.org
AlfaLinux - http://alfalinux.sourceforge.net/alfaeng.php3
Basic Linux - http://homepages.ihug.co.nz/~ichi/baslinux.html
Hal91 - http://home.tu-clausthal.de/~incp/hal91/ **RECOMENDADA**
MuLInux - http://mulinux.nevalabs.org/

/* Fin comentario */

-----------------------------------------------
#2.2 OpenBSD:

Es un sistema operativo libre, basado en BSD4.4.
Los origenes de este SO se remontan del año 1980, el proyecto fue apoyado por la
Universidad de Berkeley en California en donde el codigo fuente a sido expuesto
y cientos de programadores de todo el mundo lo analizan.
Es un SO muy nuevo.

OpenBSD tiene un enfoque diferente que el resto de los sistemas ya que esta
orientado hacia la correcta escritura de software, la auditoria continua, y la
criptografia integrada.
Es entonces, un sistema muy estable (como todos los UNIX), pero es mas eficiente
en cuanto a su desempe~o.

Su instalacion es rapida y facil, ademas cuenta con manuales muy bien elaborados.
Es de libre distribucion, modificacion y copia.

Caracteristicas en general:
**************************

- Funciona en muchas plataformas (Sparc, Intel, Alpha, etc)

- Esta reconocido por muchos programadores de todo el mundo como el SO mas
  seguro.

- Tiene todas la funcionalidades de un Unix
```

- Ofrece al usuario la posibilidad de participar en su desarrollo y prueba

- Util para la privacidad, integridad de la informacion y transmicion de datos

- Incluye muy buena documentacion con ejemplos y recomendaciones

- Muy bajo costo (gratis por internet o por medio de un CD en Internet a un precio bajo)

#2.3 FreeBSD (www.freebsd.org):

Este sistema operativo derivado de BSD-Unix, soporta arquitecturas x86, DEC Alpha, y PC-98. El soporte para otras arquitecturas esta en desarrollo.
Sus raices derivan del SO escrito por el Computer Systems Research Group de la Universidad de California, Berkeley, 4.4BSD.

Este sistema tiene buenas prestaciones en cuanto a comunicaciones de red, seguridad y compatibilidad, que segun FreeBSD inexistente en otros sistemas operativos.
Ofrece servicios de servidor para la Internet o Intranet, robustos, incluso en situaciones de carga alta, administrando de manera eficaz la memoria.
Es ideal para aplicaciones de red. Las caracteristicas para esto son:

- Compartir ficheros mediante NFS
- Distribucion de Informacion de Red con NIS
- Soporte para Logins remoros
- Soporte y configuracion remota via SNMP
- Servidor de ficheros FTP
- Resolucion de nombres de maquina con DNS/BIND
- Rutear paquetes entre multiples interfaces, incluyendo lineas PPP y SLIP
- Servicios IP Multicast (MBONE)

Podes hacer que tu PC sea un servidor web y de new, ya que trae el software adecuado para esto. Tambien podes utlizar SAMBA para compartir archivos con el amigo Win98/NT, y demas familiares de Microsoft.
Los protocolos de red Aplletalk y Novell en modo cliente/servidor son soportados.

Puede servir para estos servicios:

- WWW
- Proxy WWW
- FTP
- Ficheros e impresion

En cuanto a la seguridad FreeBSD esta muy comprometido. Incluye en el kernel soporte para firewall IP, asi tambien como Gateways e IP Proxys.
Software de encriptacion, shells seguras, kerberos, etc.

Caracteristicas avanzadas:

- Sistemas con mas de 16 megabytes operan mas eficientemente con perifericos DMA en el bus ISA.
- Los programas reciben un mejor manejo de la memoria, liberando al admin. el trabajo de ajustar los tamaños de los caches.
- Modulos de compatibilidad, que permiten ejecutar programas de otrso SO en FreeBSD, incluyendo Linux, SCO, NetBSD y BSDI
- Modulos de kernel de carga dinamica, que permite tener acceso a nuevos sistemas de ficheros, protocolos de RED, o emuladores de binarios en tiempo de ejecuion sin necesidad de generar un nuevo kernel.
- Librerias compartidas, que reducen el tema~o de los programas, ahorrando espacio en disco y en memoria.

#2.4 Solaris

Este Sistema operativo nos brinda caracteristicas de portabilidad, escalabilidad, interoperabilidad y compatibilidad en general:

* Portabilidad:

El software que esta conformado por una ABI (apalicacion de interface binaria) ejecuta un Shrink-Wrapped (contraccion envuelta) el software en todos los sistemas vendidos con la misma aruqtecturas del microprocesador.

* Escalabilidad:

Las aplicaciones se usan con mas frecuencia en el sobre tiempo, pero requiere equipos mas potentes para soportarlo.El software debe ser capaz de ejecutar en un rango de ancho poderoso y debe ser capaz de tomr ventajas del poder adicional que se esta ejecutando.

* interoperabilidad:

Solaris puede iteroperar con unos sistemas muy populares hoy en el mercado, y aplicaciones que se ejecutan en Unix se pueden comunicar muy facilmente.

* Compatibilidad:

Permanece en el ambito competitivo para minimizar sus costos y maximizar sus ingresos.

Estas serian caract. generales de Solaris, pero cada integrante de la PC tiene la suya (usuarios y administrador).

Caracteristicas para usuario:

* Espacio de trabajo: permite que los usuarios, desde una ventana hagan el manejo de los servicios en forma rapida y les permite entallar su espacio de trabajo a sus necesidades personales.

* Integracion de servicios Desktop: Incluye Drag and Drop (arrastrar y solar), Cute y Paste (copiar y pegar), proporcionando la base para que las aplicaciones puedan integrarse.

* Bibliotecas graficas: incluye XGL, Xlib, PEX, y XIL, dando soporte para aplicaciones 2D y 3D.

* Herramientas de Imagen: permite cargar, ver y guardar imagenes de 40 formatos diferentes (GIF, TIFF, JFIF, etc)

Caracteristicas para administrador:

* Sistemas de administracion de archivos: permite a los admin. mover montar, desmontar, etc, sistemas de archivos

* Manejo de procesos: permite tenes un control de los recursos del sistema, ubicacion de acceso a discos, entradas al sistema, etc.

* Usuarios y manejo de grupos: permite todo tipo de configuracion referida a los usuarios (asignar ID´s, grupos, etc)

* Seguridad: el ASET (Automated Security Enhancement Tool) que permite a los admin. configurar los permisos de archivos, y demas temas en cuanto a la seguridad de los datos del disco.

El software en Solaris es entrgado en forma de paquetes (coleccion de archivos y

directorios reuqridos para el funcionamiento del producto). Un cluster es una colecion de paquetes, y hay cuatro tipos:

* Nucleo de soporte del sistema (Core System Support): es el software de configuracion minima, y permite arrancar el sistema y ejecutar el ambiente oprativo de Solaris.

* Sistema de Soporte para Usuarios Finales (End User System Support): contiene el nucleo de soporte del sistema mas el soprte del sistema para usuarios finales, como es el Open Windows (sistema de ventanas), etc.

* Soporte de Sistemas Desarrollados (Developer System Support): Contiene soporte de usuario final del sistema más librerías, incluye archivos y herramientas que se necesitan para desarrollar el software en el sistema de Solaris. Compiladores y depuradores no están incluidos en el sistema de Solaris 2.5.

* Distribucion entera: contiene todo el ambiente Solaris.

En cuanto a redes, el sistema contiene el SAF, que es una utilidad para administrar terminales, modems y demas dispositivos de red.

Caracteristicas de SAF:

* A~adir y administrar ttymon and listen monitores en puertos (usando el comando sacadm)

* A~adir y administrar ttymon servicios de monitores en puertos (usando los comandos pmadm y ttyadm)

*A~adir y administrar listen servicios de monitores en puerto (usando los comandos pmadm y nlsadmin)

* Administrar y troubleshoot de dispositivos TTY.Administrar y troubleshoot entradas de requisitos de red para servicios de impresión.

* Administrar y troubleshoot el controlador de acceso al servicio (Service Access Controller) usando el comando sacadm.

#3 Conlusion:

Los sistemas operativos derivados de Unix cumplen un papel muy importante en la computacion, mas precisamente en servidores de Internet y, en el caso de Linux en el escritorio, ofreciendo mucha seguridad, establidad y el respaldo de miles de programadores en el mundo.
Muchas empresas dirigidas al hosting de sitios web, servicios de e-mail, de noticias y de base de datos lo elijen, puede ser por su costo (gratis, aunque esta no sea su cualidad principal) o por su libertad...
En el futuro todo tiene que ser libre, para que el usario final puede sentir que hace lo que quiera con sus cosas y no tener que "depender" de alguien en especial, asi creando un monopolio :).

#4 Bibliografia:

Toda la informacion escrita en este articulo fue obtenida de sitios de Internet y de manuales de sistemas operativos.

#5 Despedida:

Bueno... espero que este articulo les sirva para comprender mas las caracteristicas de los SO´s y su crecimiento a traves de los a~os.
Me hubiese gustado hablar sobre otros Unix´s pero el tiempo es muy escaso por estos

lados :) y no queria demorar mas esta parte.

Espero para la proxima hacer un articulo referido a las distintas licencias que existen en los computacion (referidas al software), y tambien sobre el movimiento GNU y la Fundacion del Software Libre.

Cualquier comentario me escriben a ekstor@yahoo.com.ar.

SALUDOS

KSTOR <Argentina>

EOF

-[0x06]----------
-[Curso de C]----------
-[by Lindir]----------SET-29--

The Ancient Art Of C Programming.

Introduccion.

Ultimamente veo en el foro de SET que hay muchas personas preguntando sobre como programar tal o cual cosa en C/C++. Me alegra que, aunque el tema no sea exactamente hack, sean mensajes constructivos. Estas preguntas, ademas de que aun no he visto ningun documento para iniciarse en el mundo de la programacion que me pareciera a la vez sencillo y completo, me han decidido a escribir el siguiente articulo. Espero que mi poca experiencia y mi mucho interes sean clarificadores para todo aquel que desee comenzar a programar.

He elegido el lenguaje C para hacer esto por varios motivos, que son totalmente subjetivos y que expongo a continuacion:

- -Es un lenguaje sencillo (en su descripcion)
- -Es un lenguaje actual y util
- -Es un lenguaje historico
- -Es elegante
- -Es de medio-bajo nivel pero puede servir para aplicaciones de alto nivel
- -Es compilado, y existen compiladores GRATIS y BUENOS.
- -Es el que mas me gusta, el que mas conozco y en el que tengo mas experiencia.

La programacion que voy a tratar aqui es una programacion de micro-procesadores. Los microprocesadores son unos circuitos estupidos que solo saben hacer algunas cosas:

- -Leer datos desde una zona de memoria
- -Guardar datos en una zona de memoria
- -Sumar, restar, multiplicar y dividir numeros
- -Comparar numeros y comprobar condiciones aritmetico-logicas (es cero, hay acarreo, es negativo, etc.)
- -Segun los resultados, decidir cual es la siguiente operacion a realizar

Todo esto puede parecer trivial, pero lo escribo para que se tenga claro que un ordenador no es inteligente. Si a una persona con conocimientos minimos de aritmetica y logica le dieran tiempo y papel suficiente, seria capaz de hacer lo mismo que un ordenador: seguir las ordenes del programa. Y obtendria el mismo resultado. Lo que hace a los procesadores tan potentes es que son infinitamente mas rapidos que las personas. Pero no son inteligentes. No hay tarjetas inteligentes ni casas inteligentes. Hay tarjetas programadas y casas programadas. Y si el programador no es inteligente, menos aun lo seran las tarjetas y las casas que el programe.

Con este curso intentare hacer pensar a las personas que desean aprender a programar, dar una base de algoritmia, de sintaxis de C y -puede que al final- algo de idea de arquitectura de ordenadores. Espero que no me haya fijado una tarea mas alla de mis posibilidades.

1. Los datos.

Las computadoras manejan datos. Es su unico fin. ¿Que son los datos? Los datos son trozos de informacion. Un dato puede ser mi edad, mi nombre, mi apodo (lindir), el numero de segundos que ha pasado desde las 00:00:00 del 1 de enero de 1970, si la unidad de disco duro esta ocupada, la direccion de memoria donde esta almacenado el caracter que ocupa la posicion superior izquierda de la pantalla, el valor digitalizado de la tension de salida del

microfono, etc.

1.1 Bits.

Debido al origen electronico de los procesadores y a la simplicidad del sistema binario, los datos se almacenan en bits. Un bit es un apocope del termino ingles BInary digiT. Un bit es la menor informacion que podemos almacenar, y su valor puede ser cero logico o uno logico. Segun el significado que queramos darle, podemos ver algunos ejemplos.

	Dato	Significado 0	Significado 1
a)	Sexo del usuario	Varon	Mujer
b)	Estado del boton derecho del raton	Pulsado	No pulsado
c)	Estado del pixel superior izquierdo de la pantalla en una pantalla monocroma	Apagado	Encendido

1.2. Numeros.

El bit como informacion es muy pobre. Por ello usualmente se agrupan formando octetos (bytes). Un octeto son ocho bits. Puede ocurrir que cada bit de un octeto indique una condicion binaria como las vistas anteriormente (del tipo verdadero/falso), que haya agrupaciones de varios bits (2, 3...) o que todo el octeto guarde solo un dato. El ejemplo mas directo es el octeto que almacena un numero.

Los numeros se almacenan en octetos en distintos formatos. El mas sencillo es el numero entero no negativo puro. Puesto que un octeto son ocho bits, se hace algo parecido a lo que se hace en numeracion arabiga en base 10. Cuando tenemos un numero de varias cifras, la primera se multiplica por 1, la segunda por 10, la tercera por 100... y al final se suman todos los resultados. De este modo, 326 = 3*1 + 2*10 + 6*100

En el caso que estamos viendo, el formato binario es el siguiente: el bit menos significativo almacenar un valor 0 aritmetico (cero logico) o 1 aritmetico (uno logico). Este valor se multiplica por 2**0=1 (** representa el operador exponenciacion). El segundo bit menos significativo almacena un valor que se multiplicara por 2**1=2, el tercero por 2**2=4 ,etc. De esta forma, si representamos de izquierda a derecha los valores logicos almacenados en los bits de un octeto, podemos tener los siguientes ejemplos

	Numero	Octeto binario
a)	1	00000001
b)	2	00000010
c)	5	00000101
d)	9	00001001
e)	15	00001111

Las operaciones serian:

```
a)      1 =     0 * (2**7) + 0 * (2**6) + 0 * (2**5) + 0 * (2**4) +
                0 * (2**3) + 0 * (2**2) + 0 * (2**1) + 1 * (2**0)
b)      2 =     0 * (2**7) + 0 * (2**6) + 0 * (2**5) + 0 * (2**4) +
                0 * (2**3) + 0 * (2**2) + 1 * (2**1) + 0 * (2**0)
c)      5 =     0 * (2**7) + 0 * (2**6) + 0 * (2**5) + 0 * (2**4) +
                0 * (2**3) + 1 * (2**2) + 0 * (2**1) + 1 * (2**0)
d)      9 =     0 * (2**7) + 0 * (2**6) + 0 * (2**5) + 0 * (2**4) +
                1 * (2**3) + 0 * (2**2) + 0 * (2**1) + 1 * (2**0)
e)      15 =    0 * (2**7) + 0 * (2**6) + 0 * (2**5) + 0 * (2**4) +
                1 * (2**3) + 1 * (2**2) + 1 * (2**1) + 1 * (2**0)
```

Esto es solo una forma de almacenar numeros en un octeto. Con este metodo podemos almacenar valores entre 0 y 255. Para almacenar valores mayores pueden usarse dos o mas octetos, y hay otros metodos para valores con signo o con decimales, como signo-magnitud, complemento a 1, complemento a 2, reales en coma fija o los estandares IEEE para numeros en coma flotante que mezclan los anteriores.

Como referencia, decir que la mayor parte de los procesadores modernos utiliza la representacion en complemento a 2 (Ca2) para numeros enteros y los estandares IEEE para coma flotante. No se utiliza la representacion en coma fija, pero se puede "emular" la misma con enteros en complemento a 2 y desplazamientos de bits. Si no entendeis nada de esto, no os preocupeis: lo comprendereis cuando os haga falta.

Lo importante de esto es ver como con un sencillo bit, que puede representarse electricamente como una tension alta o una tension baja, agrupando varios podemos representar cualquier dato numerico que deseemos. Esto es cierto en forma aproximada, ya que no pueden representarse numeros reales como e o pi de forma exacta. Pero nadie necesita eso en la vida real.

1.3. Caracteres.

Ya sabemos como se representan los numeros. Pero... ¿y las letras?. Muy sencillo. Para representar una letra, asignaremos a la misma un numero. De esta forma prodriamos asignar el 1 a la A, el 2 a la B, etc. Y, usando el metodo de un octeto anteriormente visto, almacenar hasta 255 letras. Pero el alfabeto tiene menos de 30 y ademas para escribir hace falta tener ciertos signos de puntuacion.

Por ello se usan representaciones estandar como el ASCII. El ASCII (American Standar Code for Information Interchange) no es mas que una relacion entre numeros y caracteres. La tabla ASCII asocia un numero a cada caracter (letra, puntuacion, espacio, tabulador, etc.). En la tabla ASCII el caracter a tiene asociado el numero 97, y el A el 65. Entonces tener almacenado en memoria la palabra ABBA en ASCII seria equivalente a tener en memoria el conjunto de numeros 65 66 66 65 o en binario:

```
        01000001 01000010 01000010 01000010
```

Existen otras representaciones alternativas (tablas de codigos) para los caracteres, pero hoy en dia la tabla ASCII es la mas utilizada por los "paises occidentales".

1.4. Tipos basicos de datos en C.

C ofrece un conjunto limitado de tipos de datos basicos, que son caracteres, numeros enteros con y sin signo y numeros flotantes. A los tipos de datos enteros (int y char) se les puede aplicar el calificador "signed" o el calificador "unsigned", indicando en cada caso si es un valor con signo (positivo o negativo) o sin signo (no negativo).

El tipo "char" ocupa un octeto, y sirve para almacenar caracteres, o tambien numeros pequeños de entre 0 y 255 (sin signo) o entre -128 y 127 (con signo).

El tipo "int" ocupa distinto segun la maquina para la que se compile el programa. En un 486/Pentium I/Pentium II sera de 32 bits. Podemos añadir el calificador "short" para conseguir un entero corto (16 bits en estos procesadores) o "long" para conseguir un entero largo (32 bits en estos procesadores). Tened cuidado si pensais crear programas portables y tratais con los octetos que forman los numeros enteros por separado: en algunas maquinas los enteros son "little endian" y en otros "big endian".

Existen dos tipos de numeros reales en C: el tipo "float" (numeros en coma flotante) y el "double" (numeros en coma flotante de doble precision). Tambien existe el tipo "long double" para numeros de coma flotante con precision extendida. No hablo mas este tipo de datos porque tampoco seran tan importantes para comenzar a programar en C. Cuando os hagan falta, podreis mirar los formatos de coma flotante del IEEE en cualquier documento de internet.

C no define ningun tipo basico booleano (verdareo/falso), pero en las expresiones logicas considera un valor 0 como falso y cualquier valor distinto de 0 como verdadero. Si el compilador genera un valor "verdadero" ,por ejemplo mediante la expresion !0 (NOT FALSE), este valor numerico sera uno. Es decir, !0 es igual a 1.

Existe un tipo de datos basico especial que son los apuntadores o punteros. Un puntero no es mas que un tipo de datos que almacena la direccion en memoria de otro tipo de datos. Asi podremos tener punteros a caracteres, a enteros, a flotantes, y el puntero especial al tipo void, que se utiliza como puntero generico (ya veremos cuando hay que usarlo). Los punteros son muy utilizados en C, pero su tratamiento lo veremos mas adelante, cuando tengamos mas idea del resto del lenguaje.

- Conversion de tipos.

Suele ocurrir que necesitamos que un valor con un tipo determinado pase a ser de otro tipo. Por ejemplo, porque ambos formen parte de una expresion. Para ello, existe conversion implicita de tipos que el compilador proporciona. Asi, en la expresion:

```
2L + 'a'
```

el valor 'a' es de tipo char, mientras que el valor 2L es de tipo long. El compilador "promociona" el tipo menor (char en este caso) al mayor (long) y el resultado sera del tipo mayor (long). El resultado de dicha expresion seria un valor de tipo long igual a 99 (recordar que 'a' en ASCII es 97). Estas conversiones de tipo son automaticas y el programador no tiene que preocuparse por ellas.

Existe otra clase de conversiones, la conversion explicita o "cast". Los casts se utilizan cuando deseamos que un tipo determinado se interprete como otro tipo. Esto es especialmente util en el caso de los punteros, como luego veremos. De cualquier modo, si utilizamos un tipo incorrecto (por ejemplo como parametro para una funcion) el compilador NOS DEJARA HACERLO, aunque posiblemente nos indique esta situacion con un aviso o "warning". Si realmente deseamos hacer eso (no es un error que se nos ha colado), podemos evitar el warning con un cast. Por ejemplo podemos hacer esto:

```
int numero = 97;
char a = (char) numero;
```

El cast es un operador unario cuya sintaxis es "(tipo_al_que_convertir)". En el ejemplo, se realiza un cast sobre la variable numero (de tipo int) al tipo char.

1.5 Comentarios.

C permite dos tipos de comentarios en el codigo de los programas. El primero es el original de C, y consiste en usar las secuencias /* y */ para englobar el texto. De esta forma, un comentario es:

```
/* Esto es un comentario */
```

Estos comentarios pueden ocupar mas de una linea y no pueden contener las secuencias limitadoras por razones obvias:

```
/* Esto es un comentario
```

```
           de varias lineas */
```

Esto ultimo significa que los comentarios no pueden estar anidados. No pueden incluirse comentarios dentro de comentarios. Por lo tanto lo siguiente es erroneo:

```
        /* Comentario padre /* Comentario hijo */ */
```

El preprocesador veria un comentario con contenido "Comentario padre /* Comentario hijo" y un finalizador de comentario "*/" sin el iniciador "/*" correspondiente.

El otro tipo de comentarios esta tomado de la sintaxis de C++ y son comentarios de una linea, usando la secuencia //:

```
        // Esto es un comentario de una linea.
```

Los comentarios no generan codigo ni reservan espacio y son totalmente eliminados por el preprocesador. Su uso es exclusivo para el mantenimiento del codigo: usad comentarios para documentar el codigo que escribais, por si mas tarde teneis que volver a entenderlo para modificarlo o por cualquier razon. No useis comentarios tontos como:

```
        if (a == 1){    /* si a vale 1... */
                ...
```

Sino algo mas bien como:

```
        /* dia_semana: funcion que toma como parametro la fecha y hora en
        segundos desde "La Epoca" y devuelve el dia de la semana (1 a 7
        comenzando por el lunes) */
        int dia_semana(int fecha){
                ...
```

1.6 Constantes.

Los datos que no cambian se denominan datos constantes, y los que si cambian se llaman datos variables.

Las constantes son datos de nuestro programa que no deben cambiar. A veces tambien se conocen con el nombre de "literales". Las constantes tambien tienen su tipo, y pueden ser enteras, de coma flotante, de caracter, de cadena de caracteres y el caso especial de las enumeraciones.

En C, las constantes enteras pueden ser de tipo int o long. Ademas, pueden ser signed o unsigned. Si solo se usa el valor numerico, el tipo sera int. Si queremos que sean unsigned, debemos añadir una u o U. Para conseguir una constante "long int" debemos hacer que termine en l o L. Ejemplos:

```
        123             Tipo int
        123L o 123 l    Tipo long int
        123U o 123 u    Tipo unsigned int
        123UL o 123 ul  Tipo unsigned long
```

Ademas, podemos utilizar representaciones en base 10 (123) en hexadecimal si comenzamos por "0x" (0x7b o 0x7B) y en octal si comenzamos por "0" (0173). Mucho cuidado que 066 es distinto de 66.

Asimismo las constantes en coma flotante pueden escribirse con un punto decimal (1.7) o con mantisa-exponente (2.8e-7 o 2.8E-7). El tipo por defecto para estas constantes es double. Si queremos una constante float, debemos terminar con F o f (1.2f o 1.2F) y si la queremos long double, con L o l (1.2L o 1.2l).

Las constantes de caracter se encierran entre signos ’’. Asi, el caracter a se escribe ’a’. Tambien pueden usarse las representaciones ’\o141’ (en octal) o ’\x61’ (en hexadecimal). Asimismo, existen los denominados

"caracteres de escape", como el de nueva linea ('\n'), la campana ('\b'), el tabulador ('\t'), etc. Recordar que el caracter \ debe escaparse a su vez, y debe ser escrito '\\'.

Las constantes de cadena estan formadas (como todas las cadenas en C) por una secuencia de octetos terminados por un octeto a cero (caracter '\0'). La forma de escribir una constante de cadena es "Esto es una constante". Para incluir el caracter '"' en una constante de cadena, hay que escaparlo, es decir, la constante con valor 'Esto es una "constante"' se debe escribir como "Esto es una \"constante\"".

C permite (gracias a su preprocesador) asociar identificadores a las constantes mediante la directiva #define. De esta forma, podemos por ejemplo escribir:

```
#define CADENA "Esto es una cadena" /* Constante de cadena */
#define EDAD_MINIMA 18  /* Constante entera */
#define EURO 166.386    /* Constante coma flotante */
```

Notar que los #define no son sentencias para el compilador (sino para el preprocesador) y por lo tanto no terminan con ; como ocurre con las demas.

Por ultimo, se permite una clase de constante numerica especial: las enumeraciones. Una enumeracion es una lista de valores enteros asociados a un nombre. Por ejemplo, para definir la enumeracion de nombre "dias", que asocie a los identificadores "LUNES", "MARTES", etc. los numeros 1,2... debe escribirse:

```
enum dias { LUNES=1, MARTES=2, MIERCOLES=2, JUEVES=4, VIERNES=5,
                SABADO=6, DOMINGO=7};
```

O, mas corto:

```
enum dias { LUNES=1, MARTES, MIERCOLES, JUEVES, VIERNES,
                SABADO, DOMINGO};
```

Si se omite el primer valor, la enumeracion comienza por cero. La siguiente enumeracion asocia el valor 0 a NADA y el valor 1 a TODO:

```
enum cantidad { NADA, TODO };
```

Alguien puede preguntarse para que vamos a necesitar una constante o enumeracion, y si no estariamos trabajando el doble al usarlas, ya que podemos obviarla y sencillamente utilizar la constante numerica directamente. Bien, hagamos pensar a ese alguien. Supongamos que tenemos un programa en el que asociamos a cada dia de la semana (comenzando por lunes) los numeros del 1 al 7. Supongamos tambien que el programa realiza copias de seguridad de los archivos los domingos. Entonces escribimos en C:

```
if (dia==7) hacer_bakckup();
```

Si guardamos ese programa y lo olvidamos hasta pasados unos años, al volver a leer nuestro codigo, ese "7" no nos da ninguna informacion: puede ser que se refiera al domingo, o al dia 7 de cada mes, o al septimo dia del año, o al septimo dia desde que el programa comenzo a funcionar... Es lo que se conoce como un "numero magico". Ahora supongamos que usamos las enumeraciones. El codigo seria:

```
if (dia==DOMINGO) hacer_backup();
```

En este caso no hay lugar a dudas. Si alguien aun no se ha convencido de que el uso de constantes con identificador y enumeraciones ahorra tiempo, que no las use: cuando realice un proyecto de programacion medianamente complejo (y si es posible con varios colaboradores) aprendera que usar numeros magicos directamente en lugar de #define y enumeraciones suele derivar en una amputacion de gonadas previo dolor de cabeza.

1.7 Variables.

Por variables entendemos las posiciones de memoria que almacenan datos que cambian durante nuestro programa (datos variables).

Para referirnos a cada una de las variables de nuestro programa les asignamos un identificador, al igual que ocurria con las enumeraciones. Los identificadores validos se componen de letras, numeros y el caracter _ y deben comenzar por una letra o por un _. Por supuesto la lengua de Cervantes queda como siempre excluida dentro del estandar, asi que no intenteis utilizar tildes ni la letra ñ en vuestros identificadores.

Todas las variables de nuestro programa deben ser declaradas. La declaracion sirve para indicarle al compilador que queremos usar una variable con un tipo y un identificador determinados. De esta forma, el compilador reserva el espacio necesario para ella, y asocia esta zona de memoria a todas las operaciones que realicemos en las que utilicemos dicho identificador.

Como norma general, suelen reservarse los identificadores con todas las letras en mayuscula para las constantes, pero C no impone esto: es solo una regla de estilo. Lo que si que no conviene es usar identificadores del tipo __NOMBRE__, _MIVARIABLE o __MICONSTANTE, puesto que suelen ser usados por las bibliotecas, salvo que estemos escribiendo una biblioteca nosotros mismos, claro esta. Hay quien gusta de mezclar mayusculas y minusculas en los nombres de variables (EstoEsMiVariable o DiaDelMes) quien usa _ para separar palabras (esto_es_mi_variable o dia_del_mes) y quien no usa nada de esto (mivariable o diames). Si no teneis claro que estilo usar, podeis buscar por internet especificacion de estilos (el estilo Kernighan y Ritchie, el GNU, etc.).

La declaracion de variable es: tipo nombre [= valor];. La parte "= valor" la escribo entre corchetes porque es opcional. Tambien puede usarse la coma para declarar multiples variables de un mismo tipo asignandoles un valor inicial a cada una, a algunas o a ninguna. Por ejemplo:

```
int dia;
int dia=1;
int dia=1,mes=1,anno=2004;
char letraA = 'A', opcion;
```

1.8 Operadores.

- Operadores aritmeticos.

Para trabajar con las constantes y variables se utilizan los operadores. Los primeros en los que alguien piensa son los operadores aritmeticos, asi que veremos cuales nos permite C.

Antes que nada, decir que los espacios entre operandos y operadores pueden obviarse, es decir, "1+1" equivale a "1 + 1". Esto ocurre en general para todos los caracteres de espacio de un programa: solo sirven para que el codigo sea claro, puesto que lo primero que hace el preprocesador es eliminar dichos caracteres (siempre que no formen parte de una constante de tipo caracter o de cadena de caracteres, en tal caso se mantienen, por supuesto). En ejemplos anteriores habreis podido notar que a veces utilizo espacios y otras veces no; es indiferente.

Los operadores aritmeticos basicos son suma (+), resta (-), multiplicacion (*), division (/) y resto de la division entera o "modulo" (%). Todos ellos pueden usarse con cualquier tipo numerico salvo %, que por razones obvias no puede usarse con numeros en coma flotante. Un ejemplo del uso de estos operadores seria:

```
int a=5,b=2;
int c=a*b;              /* c vale 10 */
int d=a/b;              /* d vale 2 */
```

```
        int e=a%b;              /* e vale 1 */
```

Existen operadores unarios (toman solo un operando) para el signo. Son + (positivo) y - (negativo). Entonces podemos escribir:

```
        a = -2;         /* a vale -2 */
        b = +5;         /* b vale 5 */
```

Tambien existen operadores unarios para incremento o decremento de variables. Son ++ (incremento en 1) y -- (decremento en 1) y pueden usarse como operadores sufijos o postfijos. Si los usamos como prefijos, i.e. ++numero, primero se realiza el incremento/decremento y luego se evalua la expresion al completo. Si se utilizan como postfijos, numero++, primero se evalua la expresion y luego actua el operador. Asi:

```
        int numero = 5;
        int numero2 = ++numero;         /* numero2=6, numero=6 */
        int numero3 = numero--;         /* numero3=6, numero=5 */
```

La precedencia de los operadores aritmeticos es: primero los operadores unarios, luego los de multiplicacion (*, / y %) y finalmente los de suma (+ y -). Pero pueden usarse los parentesis para conseguir realizar los calculos en el orden deseado. Por ejemplo, la expresion 1+2*3 es igual a 7, pero si escribimos (1+2)*3 entonces el resultado es 9.

La regla general es que los operadores de multiplicacion preceden a los de suma y respecto a los operadores logicos, && precede a ||. De cualquier modo, mejor buscad una lista con los operadores y su precedencia. Hay miles por internet y no os voy a escribir todo, ¿no? Ah, y cuando tengais una duda lo mejor es usar parentesis, que son "gratis" y no pasa nada si son redundantes.

- Operadores logicos.

Por otro lado tendremos los operadores logicos. Estos son && (AND), || (OR) y ! (NOT). Estos operadores utilizan el algebra de Boole, asi que si alguien tiene alguna duda de como funcionan, que se documente sobre el tema. Lo que debemos saber es que las expresiones logicas se evaluan de izquierda a derecha, y que ademas las subexpresiones logicas no se evaluaran si ya se sabe el resultado final de la expresion. Supongamos que tenemos dos variables de tipo entero a=1 y b=0. Entonces la expresion: (a && b) && (a || b) sera evaluada de la siguiente forma:

1. a&&b es 1&&0 = 0 (FALSO)
2. El resultado final es 0 (FALSO).

Vemos que no se ha evaluado la subexpresion a||b porque no hacia falta para hallar el resultado final: FALSO AND X es siempre FALSO, no importa el valor de X. En cambio, si b vale tambien 1, (a&&b)&&(a||b) se evaluaria:

1. a&&b es 1
2. a||b es 1
3. 1 && 1 es 1. El resultado final es 1.

En este caso si ha sido necesario evaluar a||b puesto que el resultado de 1&&X depende del valor de X.

- Operadores relacionales.

Los operadores relacionales o de comparacion permiten determinar si un numero es mayor, menor o igual que otro. Los operadores son > (mayor que), >= (mayor o igual que), < (menor que), <= (menor o igual que), == (igual a) y != (distinto de). Todas las siguientes expresiones devuelven un valor verdadero: 2>1, 2<3, 2>=2, 2<=3, 2==2 y 1!=2.

- Operadores de bit.

Finalmente, los operadores de bit actuan sobre todos los bits de un valor. Estos operadores siguen tambien el algebra de Boole a nivel de bit, y son & (AND de bit), | (OR de bit), ~ (NOT de bit) y ^ (XOR de bit). Estos operadores suelen utilizarse para "activar", "desactivar" o "testar" bits dentro de un grupo de bits (octeto, palabra de 16 bits, palabra de 32 bits, etc.) utilizando mascaras. ¿Como es esto?. Bien, supongamos la siguiente declaracion:

```
int numero = 5;
```

¿Como podemos saber si 5 es par? Bueno, direis que ya lo sabemos, pero supongamos que no iniciamos la variable numero nosotros, sino que recibimos su valor por teclado (mas tarde aprenderemos a hacer eso). Hay dos maneras sencillas de ver si un numero es par:

1. Hallar el resto de la division por 2, es decir, evaluar numero%2 y comprobar si es cero.
2. Evaluar el estado del LSB (Least Significant Bit, bit menos significativo) del numero y comprobar si es cero.

La segunda forma sera la que utilicemos. Si no comprendeis por que el LSB de los numeros enteros impares esta a 1 (activo), simplemente pensad que comienza valiendo 0 y se va alternando en cada siguiente numero:

```
Numero  LSB
0       0
1       1
2       0
3       1
... etc.
```

Bien, utilizaremos una variable de tipo int como un booleano (0 = FALSO, 1 = VERDADERO) que indicara la paridad de nuestra variable numero. El codigo completo seria:

```
int numero = 5;
int esimpar;
esimpar = numero & 1;   /* Si numero es impar, esimpar = 1 */
```

¿Que hemos hecho? Sencillamente la AND de numero con la mascara: 1. La operacion seria:

```
00000000 00000000 00000000 00000101     (5)
00000000 00000000 00000000 00000001     (1)
-----------------------------------
00000000 00000000 00000000 00000001     (5&1=1 o VERDADERO)
```

Si realizamos la AND con una mascara, el resultado tendra a 0 todos los bits que tambien esten a 0 en la mascara (bits del 1 al 31 en el ejemplo) y el resto de bits tendran el mismo valor que los correspondientes bits en el numero inicial (bit 1 en el ejemplo). Otro ejemplo para dejarlo claro puede ser: 13&9

```
00000000 00000000 00000000 00001101     (13)
00000000 00000000 00000000 00000101     (9)
-----------------------------------
00000000 00000000 00000000 00000101     (13&9=9)
```

Asimismo podemos usar la OR para activar un bit determinado o un patron de bits. Por ejemplo numero|5 activa los bits 0 y 2 de la variable numero:

```
00000000 00000000 00000000 100000001    (numero, supongamos = 129)
00000000 00000000 00000000 000000101    (5)
-----------------------------------
00000000 00000000 00000000 100000101    (129|5 = 132)
```

Y utilizar la XOR para cambiar el estado de un bit o patron de bits determinado. Por ejemplo, supongamos que queremos cambiar el estado de los bits 0 y 4 del caracter 'a' (ASCII 97). Seria entonces 'a'^17:

```
                        01100001        ('a')
                        00010001        (17)
                        --------
                        01110000        ('a'^17)
```

Podemos ver que el estado de los bits 0 y 4 ha cambiado, y que el resto de bits ha conservado su estado original.

Otro tipo de operadores de bit son los operadores de desplazamiento. Estos son << (desplazamiento a la izquierda) y >> (desplazamiento a la derecha). Si se realizan desplazamientos hacia la derecha sobre tipos unsigned, los bits mas significativos (comenzando por el de signo o MSB, bit mas significativo) se pondran a cero. Si se hace sobre cantidades con signo, se pondran a 0 en algunas maquinas y a 1 en otras, asi que mucho ojo con este caso en el que el comportamiento no esta especificado. Para los desplazamientos hacia la izquierda, siempre se rellena con bits a 0.

Los operadores de desplazamiento pueden usarse para multiplicaciones o divisiones por potencias de dos (2 elevado a tantos bits como se desplace). Asi, 5>>2 equivale a 5/(2**2) = 1. Tambien 5<<3 equivale a 5*(2**3)=40:

```
                        00000101 (5)
                        00000001 (5>>2=1)
                        00100100 (5<<3=40)
```

C no proporciona operadores para desplazamientos ciclicos, como puede ocurrir con otros lenguajes. Los bits que se "salen" del espacio de la variable simplemente desaparecen.

- Operadores de asignacion.

Durante todos los ejemplos anteriores hemos estado usando el operador de asignacion =. Este operador se usa para almacenar un valor en una variable. De este modo la siguiente sentencia almacena el valor 5 en la variable de tipo entero a:

```
        a = 5;          /* a vale 5 */
```

Tambien puede hacerse que multiples variables almacenen el mismo valor en una sola sentencia, encadenando el operador = asi:

```
        a=b=c=d=5;      /* a,b,c y d valen 5 */
```

Esto puede hacerse ya que el operador = no solo guarda el valor a la derecha en la variable de la izquierda, sino que ademas actua como una expresion que devuelve el valor almacenado. Asimismo, debido a esto la siguiente expresion se evalua como 1 (VERDADERO):

```
        1 < (a=3)
```

Suele ocurrir a menudo que utilicemos expresiones del estilo:

```
        operando1 = operando1 OPERADOR operando2;
```

Para que sea mas sencillo de escribir, se permiten los operandos de asignacion. Hay operandos de asignacion asociados a todos los operandos aritmeticos y de bit, por ejemplo +=, /=, ^=, >>=, etc. Asi:

```
        int numero = 1;
        numero +=2;             /* numero = 3 */
        numero /=2;             /* numero = 1 */
        numero ^=2;             /* numero = 3 */
```

```
        numero >>=1;            /* numero = 1 */
```

- El operador sizeof()

El compilador ofrece un operador muy util para trabajar con la memoria. El operadore sizeof() devuelve el tamaño de una variable o tipo de datos. Por ejemplo, sizeof(char) devuelve 1. Y en un Pentium, sizeof(int) devuelve 4. Otro ejemplo puede ser:

```
        short mivariable;
        int longitud;

        longitud = sizeof(mivariable);  /* longitud vale 2 en un Pentium */
```

Estos valores se calculan en tiempo de COMPILACION. Es decir, el compilador evalua el tamaño de la variable o tipo de datos y utiliza dicho valor como una constante, no se calcula en tiempo de ejecucion. Esto que ahora puede parecer oscuro quedara mas claro cuando veamos asignacion dinamica de memoria. El operador sizeof() no puede utilizarse para hallar el tamaño de una zona de memoria reservada dinamicamente. En cambio, sizeof() puede usarse para calcular el tamaño de una matriz constante, estatica o automatica, como luego veremos.

- Otros operadores.

Existen otros operadores, como la desreferencia (*), el operador direccion de (&), el operador de conversion explicita de tipos (cast) visto anteriormente, etc. Los operadores aun no vistos los iremos introduciendo conforme vayamos avanzando con el lenguaje.

1.9 Punteros y matrices.

Hasta ahora hemos tratado con tipos de datos basicos: enteros, caracteres y numeros en coma flotante. Hemos hablado algo de cadenas de caracteres, pero aun no esta claro como tratar con ellas. Ademas, hemos dejado aparte un tipo de datos fundamental en C: los punteros o apuntadores.

- Punteros.

Un apuntador o puntero no es mas que una variable que almacena la direccion de memoria de otra variable. ¿Para que puede servirnos esto? Pues para acceder a esta variable *de forma indirecta*. Ahora veremos esto con mas detenimiento.

Para declarar una variable de tipo puntero en C, hay que seguir la siguiente sintaxis:

```
        tipo_al_que_apunta * identificador;
```

Por ejemplo, vamos a declarar a continuacion tres variables de tipos puntero a caracter, a entero y a flotante respectivamente:

```
        char *pcaracter;
        int *pentero;
        float *pflotante;
```

Aunque pcaracter, pentero y pflotante son de distinto tipo, las tres son punteros. Y un puntero no es mas que un numero: la direccion de la variable a la que esta apuntando. En cada maquina, las direcciones ocupan un tamaño determinado. En los x86 modernos (nada de 8086...) de Intel son de 32 bits. Por lo tanto todos los punteros ocupan el mismo espacio en memoria, no importa a que tipo apunten.

Bien. Ya sabemos declarar un puntero. ¿Como lo utilizamos? ¿Como podemos almacenar un valor util en el? Bueno, es sencillo. Vamos a utilizar el

operador & (direccion de). &mivariable nos devuelve la direccion en la que se almacena la variable "mivariable". Y ese valor es precisamente lo que podemos almacenar en un puntero. Entonces:

```
int numero;
int *puntero_a_numero;
puntero_a_numero = &numero;
```

A partir de este instante, puntero_a_numero almacena la direccion de la variable numero. Ahora queremos acceder a la variable de forma indirecta, es decir, a traves de su direccion almacenada en el puntero. Para ello, usamos el operador * (desreferencia o indireccion). *puntero_a_numero accede a la direccion de memoria almacenada en puntero_a_numero. Podemos utilizar esto para leer dicha direccion o para escribir en ella. Por ejemplo:

```
int numeroa;
int numerob = 5;
int *punteroa=&numeroa;
int *punterob=&numerob;

numeroa = *punterob;    /* numeroa vale 5 ahora */
*punteroa = 8;          /* numeroa vale 8 ahora */
```

La orden "numeroa = *punterob;" almacena en la variable numeroa el contenido de la direccion de memoria almacenada en la variable punterob. Puesto que anteriormente (*punterob=&numerob;) hemos almacenado en punterob la direccion de memoria de la variable numerob, lo que hacemos es en definitiva equivalente a la orden numeroa = numerob;.
Asimismo, con la orden "*punteroa=8" lo que hacemos es almacenar el valor 8 en la posicion de memoria guardada en la variable punteroa. Como anteriormente hemos guardado la direccion de memoria de numeroa en la variable punteroa, la orden anterior es equivalente a numeroa = 8;.

Puede parecer que los punteros no sirven para mucho, porque los ejemplos que hemos visto son sencillos. Pero los punteros son utiles para muchas cosas, por ejemplo:

- Paso de parametros "por referencia"
- Paso de estructuras como parametros a funciones (por referencia)
- Tratamiento de matrices de tamaño variable
- Reserva dinamica de memoria y uso de dicha memoria
- Definicion y uso de estructuras complejas como pilas, listas, arboles, etc.

Todas estas funciones son demasiado complejas para explicarlas aun, pero mas tarde veremos todas y cada una de ellas. De momento, solo trataremos el uso de punteros para acceder a matrices.

Aunque en todos los ejemplos anteriores hemos utilizado el operador &, tambien es posible guardar una direccion cualquiera (en forma de un numero) en un puntero, pero en general no sabremos que es lo que hay en dicha direccion. Y si intentamos acceder a una direccion que no pertenece a la zona de memoria de nuestro programa, el sistema operativo (suponiendo un sistema de multiprogramacion como Windows o Unix) lo detectara y terminara la ejecucion del programa. Si el sistema permite acceso total a la memoria (como MS-DOS) y modificamos zonas de memoria aleatoriamente, el comportamiento del sistema puede volverse inestable, incluso colgarse o reiniciarse.

Esto ultimo debe ser meditado. ¿Que ocurre si nos equivocamos y escribimos un programa con un puntero que modifica una zona de memoria erronea? Pues que el sistema acabara nuestro programa. Es el tipico mensaje "Segmentation fault" de Unix, o el mensaje "El programa realizo una operacion no valida" en Windows. Asi que el trabajo con punteros debe hacerse de forma cuidadosa.

- Matrices.

Un tipo de datos muy utilizado en programacion es la matriz o tabla, bien unidimensional o multidimensional. Una matriz no es mas que una serie de valores a los que se accede a traves de uno o varios indices. Veamos el ejemplo mas tipico de una tabla bidimensional:

```
Tabla:               Columna
           |   1   |   2   |   3   |
         ----+-------+-------+-------+
    F     1  |   20  |   16  |   -3  |
    i     2  |    0  |  -15  |   -8  |
    l     3  |   12  |   10  |    5  |
    a     4  |    1  |    0  |   -2  |
```

Indicando la fila y columna (los dos indices, en este caso) y el nombre de la tabla, podemos saber a que elemento nos estamos refiriendo. Asi, Tabla[1,3] (indico [fila,columna]) almacena el valor -3.

La tabla mas sencilla que puede declararse en C es un vector. Un vector no es mas que una tabla de una fila solamente. Para declarar un vector de 12 elementos enteros llamado lluvias se escribe:

```
int lluvias[12];
```

El compilador reservara entonces 12 bloques consecutivos de memoria de tamaño sizeof(int). Eso equivale a un bloque de memoria de tamaño 12*sizeof(int). Lo importante es que los elementos se almacenan DE FORMA CONSECUTIVA. Luego veremos que eso nos sirve para recorrer una tabla mediante el uso de punteros.

Este vector lluvias puede utilizarse por ejemplo para almacenar el numero de litros por metro cuadrado que ha caido en la ciudad cada mes. Para acceder a cada uno de los elementos del vector, debe usarse la sintaxis lluvias[indice]. El valor indice debe estar comprendido entre 0 y 12-1=11. Por lo tanto, si en Enero (mes 0) cayeron 20 litros/m**2 podemos escribir:

```
lluvias[0] = 20;
```

Siguiendo con este ejemplo, podemos hacer algo mas elegante, que seria lo siguiente:

```
enum meses {ENERO, FEBRERO, MARZO, ABRIL, MAYO, JUNIO, JULIO,
        AGOSTO, SEPTIEMBRE, OCTUBRE, NOVIEMBRE, DICIEMBRE};
lluvias[ENERO] = 20;
```

Tambien podemos iniciar el vector en su declaracion. Para ello, hay que indicar TODOS los valores separados por comas entre {} de este modo:

```
int lluvias[12]={20, 30, 23, 45, 15, 10, 6, 3, 10, 13, 20, 22};
```

Asi, lluvias[0] vale 20, lluvias[1] vale 30, lluvias[2] vale 23, etc.

Incluso podemos obviar el tamaño del vector y solo indicar los valores de iniciacion. En tal caso, el compilador calcula el tamaño para que quepan todos los elementos de iniciacion y NINGUNO mas. Entonces la siguiente declaracion crea un vector de 5 elementos enteros llamado vector:

```
int vector[] = {1, 2, 3, 2, 1};
```

Ya sabemos como usar tablas unidimensionales. Las tablas bidimensionales son tambien muy utilizadas. Para declarar una tabla bidimensional de tipo int de 3 filas por 2 columnas de nombre tabla seria:

```
int tabla[3][2];
```

Tambien se puede iniciar la tabla asi:

```
int tabla[3][2] = { {2,4}, {5,-1}, {4,7} };
```

Hay que hacer notar que al declarar e iniciar una tabla n-dimensional, todas las dimensiones menos la primera deben ser especificadas. Podemos obviar la primera, pero no las restantes. Lo siguiente sera incorrecto y correcto, respectivamente:

```
        int tabla[][] = { {2,4}, {5,-1}, {4,7} };       /* Incorrecto */
        int tabla[][2] = { {2,4}, {5,-1}, {4,7} };      /* Correcto */
```

Un uso claro de una matriz bidimensional puede ser la definicion de un caracter de 8x8 pixels en una pantalla de escala de grises. Cada elemento de la matriz representa el nivel de gris en el punto determinado por los dos indices. Generalmente se comienza por el punto superior izquierdo y se acaba en el inferior derecho. El valor almacenado sera de tipo caracter para tener una escala de 256 tonalidades de grises. El valor 0 indica negro y el 255 blanco. Por ejemplo podemos dibujar el caracter ’a’ y ver como seria la representacion correspondiente en C en una matriz.

A continuacion un 1 indica la maxima intensidad de luz (255) y un 0 la minima (0). El caracter punto a punto seria:

```
        Matriz de puntos           Pixels en pantalla

        +-+-+-+-+-+-+-+-+
        |0|0|0|0|0|0|0|0|
        +-+-+-+-+-+-+-+-+
        |0|0|0|0|0|0|0|0|
        +-+-+-+-+-+-+-+-+
        |0|1|1|1|1|0|0|0|                 * * * *
        +-+-+-+-+-+-+-+-+
        |0|0|0|0|1|1|0|0|                       * *
        +-+-+-+-+-+-+-+-+
        |0|1|1|1|1|1|0|0|                 * * * * *
        +-+-+-+-+-+-+-+-+
        |1|1|0|0|1|1|0|0|               * *     * *
        +-+-+-+-+-+-+-+-+
        |1|1|0|0|1|1|0|0|               * *     * *
        +-+-+-+-+-+-+-+-+
        |0|1|1|1|0|1|1|0|                 * * *   * *
        +-+-+-+-+-+-+-+-+
```

Y la forma de hacer esto en C seria (uso representacion hexadecimal):

```
char caractera[][8] =   {{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                         {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                         {0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
                         {0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0x00, 0x00},
                         {0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00},
                         {0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00},
                         {0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00},
                         {0x00, 0xff, 0xff, 0xff, 0x00, 0xff, 0xff, 0x00}};
```

En casi todas las representaciones de graficos se utiliza algo asi. Esto se llama mapa de bits (de ahi la extension de los archivos BMP, Bit MaP, de Windows). Para las imagenes en color pueden utilizarse tres caracteres por punto, cada uno de los cuales indica el nivel de rojo, verde o azul de dicho punto, etc. Asi podemos ver que las tablas son algo muy utilizado en programacion.

Por supuesto, tambien pueden declararse matrices tridimensionales o n-dimensionales (con n fijo). Aunque pueda parecer que una tabla de mas de tres dimensiones no es util, las matematicas trabajan con tablas n-dimensionales, y C permite dichas tablas. Pensad por ejemplo en un documento de una hoja de calculo, en la que hay varias paginas, cada una de

ellas con varias tablas. Podemos almacenar toda esta informacion en una sola tabla si dedicamos un indice para el numero de pagina, otro para el numero de tabla, otro para la fila de la tabla y otro para la columna. Asi solo necesitamos una tabla de 4 dimensiones que C nos proporciona al instante, en lugar de usar por ejemplo 4 tablas de 3 dimensiones distintas... En fin, que realmente puede ser que os hagan falta, y ahi estaran para cuando lo necesiteis.

Hay que hacer notar que en una tabla rectangular como esta, todas las filas tienen el mismo numero de columnas. En otros lenguajes pueden definirse tablas con distinto tamaño de fila, pero no en C. Para hacer algo asi en C se utilizan tablas de punteros, como luego veremos, que ademas son mas versatiles que las simples tablas rectangulares.

Por ultimo, podemos utilizar el operador sizeof() si queremos saber cuanta memoria ocupa nuestra tabla. Por ejemplo en la tabla caractera[][] anterior, sizeof(caractera) nos devolvera un valor de 64 (8*8).

- Aritmetica de punteros.

Los punteros tienen una aritmetica muy bien definida en C. Un puntero puede hacerse apuntar a una zona de memoria, pero luego podemos incrementarlo para que apunte a otra zona. Tambien podemos restar dos punteros para calcular la distancia en posiciones de memoria que separa sus valores. Vamos a ver todo esto. Aedmas, aprovecharemos para establecer relaciones entre matrices y punteros que nos seran muy utiles.

Lo primero que vamos a ver es como utilizar un puntero para recorrer un vector de caracteres. Lo vemos en el siguiente ejemplo:

```
        char cadena[]={'H','o','l','a','\0'};
        char *pcar;
        pcar = &cadena[0];      /* *pcar vale 'H' */
        pcar += 1;              /* *pcar vale 'o' */
        pcar += 2;              /* *pcar vale 'a' */
        pcar++;                 /* *pcar vale '\0' */
```

Vemos que primero almacenamos la direccion de cadena[0] en pcar. Luego vamos incrementando pcar y en cada caso el puntero apunta a un caracter distinto de la variable cadena, segun el incremento que hayamos aplicado. Conviene comentar que la expresion cadena devuelve precisamente la direcion del primer elemento de cadena, es decir cadena es equivalente a &cadena[0]. Por lo tanto podemos sustituir pcar=&cadena[0] por pcar=cadena.

Ademas, vamos a ver otro ejemplo:

```
        int datos[] = {1, 2, 3, 4};
        int *pdat = datos;

        pdat++;         /* Ahora *pdat vale 2 */
        pdat++;         /* *pdat vale 3 */
        pdat-=2;        /* *pdat vale 1 */
```

Vemos que el compilador sabe que pdat apunta a un tipo int, por lo tanto al incrementar pdat mediante pdat++; el compilador lo incrementa hasta el siguiente entero. Es decir, no lo incrementa en 1, sino en sizeof(int). Esto es algo muy importante que puede dar lugar a errores dificiles de detectar a simple vista. Un puntero siempre se incrementa el tamaño del tipo al que apunte.

Asimismo, no hace falta incrementar un puntero para recorrer una matriz. Basta sumar un entero a un puntero y utilizar el operador desreferencia. Por ejemplo podemos hacer:

```
        int datos[] = {1, -1, 2, -2, 3, -3};
```

```
        int valor;
        int *pdat=datos;

        valor = *pdat;          /* valor vale 1 */
        valor = *(pdat+1);      /* valor vale -1 */
        valor = *(pdat+5);      /* valor vale -3 */
        valor = *(++pdat);      /* valor vale -1 */
        valor = *(pdat-1);      /* valor vale 1 */
```

Tambien podemos restar dos punteros o compararlos. Por ejemplo:

```
        int datos[] = {1, -1, 2, -2, 3, -3};
        int valor;
        int *pdat1=datos;
        int *pdat2=&datos[1];

        valor = pdat2-pdat1;    /* valor vale 1 */
        pdat2 += 3;
        valor = pdat2-pdat1;    /* valor vale 4 */
        pdat2 = pdat1;
        valor = (pdat1==pdat2);         /* valor vale 1 (VERDADERO) */
        valor = (pdat1>pdat2);          /* valor vale 0 (FALSO) */
```

- Relacion entre punteros y matrices.

Los punteros y las matrices estan muy relacionados. Principalmente porque podemos hacer con punteros todo lo que se hace con matrices. De hecho, hemos visto que si declaramos una matriz por ejemplo con "char matriz[256];", la expresion "matriz" devuelve un valor de tipo "char *" (puntero a char) que apunta al primer elemento de la matriz.

Los punteros suelen utilizarse mucho para recorrer tablas, y sobre todo son fundamentales para trabajar con estructuras de memoria dinamica, como veremos mas tarde cuando tratemos las estructuras de datos clasicas y la memoria dinamica.

Aun asi, hay una gran diferencia entre un puntero y una tabla, que a menudo no se comprende bien al principio. Veamos las siguientes declaraciones:

```
        char cadena[] = "Esto es un ejemplo";
        char *pcadena = cadena;
```

Al procesar la primera declaracion, el compilador reserva 19 (el numero de caracteres de la cadena, mas uno mas para el '\0' o terminador) caracteres de memoria. En la seguna declaracion, el compilador reserva SOLO el tamaño de un puntero (4 caracteres en 486/P/PII/PIII, etc.). Entonces, aunque al evaluar las expresiones cadena[3] y *(cadena+3) el resultado sea el mismo, la memoria reservada en cada caso no lo es. Supongamos otro caso:

```
        char cadena[] = "Esto es un ejemplo";
        char *pcadena = "Esto es un ejemplo";
```

La primera declaracion reserva una variable de 19 caracteres que pueden modificarse. De hecho la siguiente proposicion es correcta:

```
        cadena[0] = 'e';
```

En cambio, la segunda reserva una variable de tipo char* y la hace apuntar a una CONSTANTE. De forma que la proposicion:

```
        *pcadena = 'e';
```

tiene un comportamiento indefinido y (aunque el compilador puede aceptarla) no es una expresion correcta, ya que esta intentando modificarse una constante. En cambio si es correcto el siguiente codigo:

```
        char *pcadena = "Primer ejemplo";
        pcadena = "Segundo ejemplo";
```

En este caso tenemos una variable de tipo char* que primero apunta a una constante y luego apunta a otra. Sin embargo no podemos compilar el codigo siguiente:

```
char cadena[] = "Primer ejemplo";
cadena[] = "Segundo ejemplo";
```

El compilador no acepta el codigo anterior como valido, ya que no se puede hacer una asignacion a una matriz de caracteres (por ejemplo en Java y otros lenguajes esto si puede hacerse). Por cierto que si alguien se esta preguntando como se copia una cadena de caracteres en otra, dire que:

1. Es necesario que la cadena destino tenga el tamaño suficiente como para albergar a la cadena origen, terminador ('\0') incluido
2. Hay que copiar la cadena caracter a caracter hasta encontrar el terminador.

Tambien pueden usarse las funciones strcpy() y strncpy(), preferiblemente esta ultima por cuestiones de seguridad, pero lo que hacen estas funciones es precisamente una copia caracter a caracter. Luego hablaremos de funciones y de la biblioteca estandar.

Otro error comun a la hora de escribir un programa que utiliza punteros es el siguiente:

```
char *pcadena;
*pcadena='H';
*(pcadena+1)='o';
*(pcadena+2)='l';
*(pcadena+3)='a';
*(pcadena+4)='\0';
```

Podria pensarse que el codigo anterior lo que hace es ir escribiendo caracter a caracter ('\0' incluido) la cadena "Hola" en memoria, de forma que la variable pcadena apuntase a la misma. Pues no. De nuevo, el compilador *solo* reserva espacio para un char*, y no para los caracteres que componen la supuesta cadena. Asi que estamos escribiendo en una zona de memoria que, digamos, no nos pertenece. La respuesta a la hora de ejecutar ese codigo es la siguiente: Segmentation fault. Es mas, ni siquiera hemos iniciado el valor al que apunta *pcadena, con lo cual no sabemos en que direccion de memoria intentamos escribir. Conclusiones de este ejemplo:

1. Al declarar un tipo char* (o cualquier otro puntero) el compilador solo nos proporciona el espacio necesario para ese puntero, NO PARA LOS CONTENIDOS A LOS QUE APUNTA.
2. NUNCA hay que utilizar el valor de una variable que no se haya iniciado anteriormente. El valor de una variable puede iniciarse bien en el codigo (variable=2;) bien leyendolo de cualquier sitio (un archivo, el teclado, el valor de retorono de una funcion, etc.).

Puede ser que vuestro programa funcione en algunos ordenadores incluso utilizando variables no iniciadas, pero eso no significa que sea un codigo correcto: un programa correcto debe poder funcionar con independencia de las condiciones temporales o circunstanciales del sistema sobre el que se ejecute, y utilizar variables sin iniciarlas impide que esto ocurre. Mucho ojo, porque si "suena la flauta" y el programa funciona en vuestro ordenador va a ser muy dificil encontrar y corregir un fallo de este tipo. Por ello, sobre todo con punteros, conviene iniciar el valor de todas las variables en su declaracion; al menos en los comienzos hasta que domineis este tema. De cualquier modo, un compilador medianamente correcto debe indicaros esta circunstancia mediante un "warning".

- Punteros a punteros y tablas de punteros.

Alguien puede pensar que un puntero a puntero es la mayor tonteria que se

puede inventar y que es totalmente inutil. Nada mas lejos de la realidad. Las tablas de punteros, y por ende los punteros a punteros, son muy utilizados. Supongamos un ejemplo medianamente sencillo: queremos almacenar en memoria los nombres de los dias de la semana:

-Solucion 1 (en cierto modo chapucera): tener una tabla de caracteres bidimensional con 7 filas y 10 columnas, para poder almacenar los 9 caracteres de "miercoles" (el nombre mas largo) mas el terminador.
-Solucion 2 (la mas versatil): tener una tabla de punteros a caracter y hacer que cada uno de ellos apunte a la cadena que necesitemos.

La primera solucion se escribe:

```
char dias[][10]={"Lunes", "Martes", "Miercoles", "Jueves", "Viernes",
                 "Sabado", "Domingo"};
```

Y la segunda:

```
char *dias[] ={"Lunes", "Martes", "Miercoles", "Jueves", "Viernes",
                 "Sabado", "Domingo"};
```

¿Son iguales? No. ¿Por que? Sencillamente porque en el primer caso tenemos una tabla de 7x10 caracteres y en el segundo tenemos una tabla de 7 punteros a caracter, cada uno de ellos apuntando a una cadena de caracteres constante. Hay un cierto "desperdicio" de caracteres en la primera solucion.

Pero este no es el problema mas grave. En nuestro caso, sabemos que el nombre de los dias de la semana no va a cambiar. Pero si estuviesemos tratando con variables en lugar de constantes, en el primer caso habria que limitar el tamaño maximo, y este seria el mismo para todas las cadenas. En el segundo caso, podemos tener cadenas de tamaños distintos y, ademas, hacer que en un momento dado el puntero apunte a otra cadena de longitud incluso mayor sin modificar para nada la estructura de la tabla. Esto puede parecer algo oscuro ahora, pero cuando tratemos con estructuras de datos cuyo tamaño es desconocido a la hora de compilar el programa y con memoria dinamica, la ventaja de las tablas de punteros frente a las tablas bidimensionales se hara patente. El ejemplo mas claro es el de los argumentos en linea de comandos que se vera mas adelante, cuando tratemos la funcion main().

2. El flujo de programa

Hasta ahora hemos tratado de los datos, que son la materia prima para los programas. Ahora tocaremos lo que es el "estado" del programa, la evolucion del mismo en el tiempo desde que se comienza a ejecutar.

Veremos a continuacion el control de flujo del programa mediante bucles y estructuras if..else o switch..case, que son la base para la generacion de codigo util. Mas tarde hablaremos de las funciones, parametros y valores de retorno, de forma que ya estaremos en condiciones de escribir programas sencillos pero reales (obviando la entrada/salida, que sera explicada mas tarde).

Los programas se ejecutan en los procesadores de manera secuencial: una instruccion detras de otra. Realmente no hay ejecucion en paralelo como puede existir en el hardware digital (por ejemplo, pueden hacerse dos sumas a la vez con dos circuitos sumadores). Si tenemos sistemas operativos multitarea no es porque permiten ejecucion de instrucciones a la vez, sino porque ejecutan un trozo de un programa, lo paran y pasan a ejecutar otro trozo de otro. Esto lo hacen de forma ciclica y en intervalos de tiempo tan pequeños que nos parece que ambos programas se ejecutan a la vez.

2.1 Control de flujo.

En C, el flujo de programa es, por tanto, secuencial, al contrario de como puede ocurrir por ejemplo con el lenguaje de programacion de logica digital VHDL. Las instrucciones se ejecutan "de arriba a abajo". Pero a menudo hay que hacer que ciertas instrucciones se repitan un numero determinado de veces, o hasta que se cumpla cierta condicion, o que se ejecuten solo si se cumple cierta condicion... para todo ello existen las instrucciones de control de flujo.

- if ... else ...

El control de flujo mas simple que puede realizarse es el de ejecutar una instruccion solo si se cumple una condicion. Para ello se utiliza la palabra reservada if. La estructura de un bloque if es:

```
        if (condicion)
                sentencia1;
        else
                sentencia2;
```

Si la condicion se evalua como VERDADERO, se ejecutara sentencia1. En caso contrario, se ejecuta sentencia2. Por ejemplo

```
        if (numero%2==0)
                par = 1;
        else
                par = 0;
```

En este caso, si la variable numero almacena un valor par numero%2==0 se evalua a VERDADERO y se ejecuta par=1. En caso contrario, se ejecuta par=0.

Puede ocurrir que queramos ejecutar mas de una sentencia. En tal caso debemos definir un bloque de codigo. Los bloques de codigo se encierran entre caracteres { y }. En el ejemplo:

```
        if (numero%2==0){
                par = 1;
                impar = 0;
        }
        else {
                par = 0;
                impar = 1;
        }
```

Tras la ejecucion del if, si numero es par, par=1 e impar=0. En caso contrario, par=0 e impar=1.

Generalmente es una buena practica de programacion utilizar las llaves {} aunque solo sea para una sentencia dentro del if, puesto que ayuda a que el codigo sea mas claro y evita errores si luego queremos añadir mas proposiciones al cuerpo del if. Asimismo, conviene indentar (introducir espacios o tabuladores en el cuerpo del if y el else) para conseguir un codigo mas legible. Todo esto son consejos de estilo para nuevos programadores, podeis seguirlos o no, pero en principio os permitiran tener las ideas mas claras.

Si la parte de else no es necesaria, puede obviarse. Por ejemplo:

```
        if (numero==2){
                esdos=1;
        }
```

Tambien pueden utilizarse estructuras del estilo:

```
        if (numero==2){
                esdos=1;
        }
        else if (numero==3){
```

```
                estres=1;
        }
        else{
                esdos=0;
                estres=0;
        }
```

Solo uno de los tres bloques se ejecutara, aunque para condiciones tan sencillas conviene utilizar las estructuras switch/case, que luego veremos.

Por ultimo, existe un operador asociado a if, el operador ternario ?. Este operador tiene la siguiente sintaxis:

```
        condicion ? expresion1 : expresion2
```

El valor devuelto por el codigo anterior sera expresion1 si condicion se evalua a VERDADERO y expresion2 si condicion se evalua a FALSO. Asi, podemos escribir el primer ejemplo de esta seccion como:

```
        par = (numero%2==0)?1:0;
```

He escrito la condicion entre parentesis para hacer el codigo mas legible, pero no es necesario. Asimismo, suele utilizarse el hecho de que un valor 0 se evalua como FALSO para escribir las expresiones de forma compacta. Podemos escribir la sentencia anterior como:

```
        par = (numero%2)?0:1;
```

Vamos a pensar esto un poco. Si numero es par, numero%2 sera 0, es decir, FALSO. Por lo tanto el operador ternario devolvera el segundo valor y par valdra 1. Si ocurre lo contrario, devolvera el primero y par valdra 0. Podemos escribirlo tambien de una forma mas clara:

```
        par = (!(numero%2))?1:0;
```

Esto seria: si numero%2 vale 0, !(numero%2) vale 1 y par valdria 1. En caso contrario, !(numero%2) valdria 0 y par tambien. Finalmente, para dar una vuelta de tuerca mas, podemos hacerlo sin el operador ternario, simplemente:

```
        par = !(numero%2);      /* Pensadlo un poco */
```

Estos casos no son tan raros, y suele ocurrir que se utilice una variable como "llave" para entrar o no en una condicion. Asi, podemos escribir:

```
        if (llave){
                ....
        }
```

El codigo entre {} se ejecutara si llave es distinto de 0. Si escribimos:

```
        if (!llave){
                ....
        }
```

El codigo se ejecutara solo si llave es 0. Como he dicho, estas construcciones son corrientes, asi que mejor que os vayais familiarizando con ellas si pensais leer y escribir programas en C.

- while.

Otra de las cosas que se necesita a menudo es conseguir que un bloque de codigo se ejecute mientras se cumpla una condicion. Para ello estan los bucles while. Su estructura es:

```
        while (condicion)
                instruccion_o_bloque;
```

Cuando el flujo de programa llega al while, se evalua condicion. Si es VERDADERO, se ejecuta instruccion_o_bloque una vez y se vuelve a evaluar condicion. Este proceso se repite hasta que se evalue condicion y resulte

```
ser FALSO. Por ejemplo:
        int a=2;
        while (a>0){
                a--;
        }

Este ejemplo se ejecutaria:
        - a = 2         -> almacena 2 en a.
        - ¿a>0? Si      -> sigue con el bucle.
        - a--           -> ahora a = 1.
        - ¿a>0? Si      -> sigue con el bucle.
        - a--           -> ahora a = 0.
        - ¿a>0? No      -> fin del bucle.

Podemos utilizar bucles while para muchas cosas. Otro ejemplo sencillo es
hacer una multiplicacion como sumas sucesivas:
        int a=5;
        int b=2;
        int a_por_b=0;
        while (a--)
                a_por_b += b;

La condicion a-- comprueba primero si a es distinto de cero y luego
decrementa a en 1. El bucle anterior sirve para multiplicar dos numeros
mayores que (o iguales a) cero, 5 y 2 en este caso. La unica instruccion del
bucle va sumandole b a la variable a_por_b. De esta forma, el bucle suma a
(5) veces el valor b (2) a a_por_b, que es precisamente lo mismo que
multiplicar a por b y almacenarlo en a_por_b.

- do ... while.

Puede ocurrir que deseamos que se ejecute el cuerpo del while al menos una
vez, independientemente del valor de la condicion al inicio del bucle. Para
ello existe la estructura do ... while, que se escribe:
        do
                instruccion_o_bloque;
        while (condicion);

Vamos a ver un ejemplo:
        int numero = 0;
        do
                numero--;
        while(numero>0);

Al ejecutar este codigo, numero valdra -1. Si lo hubieramos escrito con un
while:
        int numero = 0;
        while(numero>0)
                numero--;

Al ejecutarlo, numero valdra 0. Nunca se entra en el bucle, al contrario que
con el do.

- for.

Los bucles for son muy utilizados para recorrer vectores y matrices, aunque
pueden usarse para muchas otras cosas. Su estructura es:
        for (proposicion1;proposicion2;proposicion3)
                cuerpo_del_bucle;

Hay que tener en cuenta que cualquier proposicion devuelve un valor, es decir,
es tambien una expresion evaluable. La ejecucion de esto es como sigue: al
```

llegar al inicio del bucle, se ejecuta proposicion1. Despues se evalua proposicion2, y si es VERDADERO se ejecuta cuerpo_del_bucle. Luego se ejecuta proposicion3. Despues se vuelve a evaluar proposicion2 y se continua de forma ciclica hasta que se evalue proposicion2 y sea FALSO. Un equivalente mediante while es:

```
        proposicion1;
        while (proposicion2){
                cuerpo_del_bucle;
                proposicion3;
        }
```

Normalmente, proposicion1 se utiliza para la iniciacion de variables, proposicion2 para la comprobacion de fin de bucle y proposicion3 para incrementos de variables indice, etc. Aunque no tiene por que ser asi, Kernighan y Ritchie advierten que es de mal estilo utilizar proposiciones que hagan otras cosas dentro de los parentesis de for.

El ejemplo de la multiplicacion puede escribirse de forma mas compacta con for:

```
        int a=5,b=2,a_por_b;
        for (a_por_b=0;a>0;a--)
                a_por_b += b;
```

Cada una de las tres proposiciones del for puede obviarse. De este modo, un bucle infinito (es decir, que nunca acaba) puede escribirse por ejemplo como:

```
        for (;;){
                ....    /* Cuerpo del bucle infinito */
        }
```

- switch ... case .... default .... break y continue.

A menudo hay que contrastar un valor numerico contra distintos valores constantes. Para ello se utiliza switch. Su estructura general es:

```
        switch (condicion){
        case CONSTANTE1:
                cuerpo1;
                break;
        case CONSTANTE2:
                cuerpo2;
                break;
        ....
        default:
                cuerpo_default;
                break;
        }
```

Veamos un ejemplo para comprender su funcionamiento. Supongamos una variable "opcion" que almacena el numero de opcion seleccionada por teclado por el usuario. Por ejemplo, la opcion 1 sumara dos numeros y la 2 los multiplicara. El resto de opciones no son correctas. El codigo puede ser

```
        switch(opcion){
        case 1:
                resultado = a+b;
                break;
        case 2:
                resultado = a*b;
                break;
        default:
                opcion_incorrecta = 1;
                break;
        }
```

La palabra clave default indica el inicio de las instrucciones a ejecutar si opcion no concuerda con ninguna de los case anteriores (es decir, es distinto de 1 y de 2). No es necesario incluir un default, y tampoco es necesario que vaya al final de todos los otros case, puede ir donde se desee. Pero si es obligatorio que solo aparezca una vez dentro del switch.

La palabra clave break sirve para terminar de forma prematura un bucle o un switch. Puede usarse tambien con for, do y while, aunque se desaconseja por no ser de buen estilo de programacion. Por ejemplo puede hacerse:

```
        for (;;){
                if (condicion)
                        break;
                .....           /* Resto del bucle */
        }
```

De esta forma conseguimos que el bucle infinito iniciado por for(;;) termine cuando se evalue condicion como VERDARERO. Todo lo que puede hacerse con un break puede hacerse sin el (salvo quizas salir de un switch) escribiendo solo un codigo mas estructurado y puede que usando alguna variable mas. Asi que, en aras del buen estilo, intentad limitar vuestros breaks a los switch.

Otra palabra reservada que permite alterar la ejecucion normal de un bucle es continue. Una sentencia continue hace que el control de flujo vuelva a ir al inicio del bucle, es decir, que el resto de la iteracion actual no se ejecute. Por ejemplo:

```
        while(a>2){
                if (b==1)
                        continue;
                ...     /* Resto de operaciones */
        }
```

Si se evalua b==1 como cierto, el resto de operaciones no se realiza y se vuelve a evaluar a>2 y comenzar una nueva iteracion del bucle. En este ejemplo se crearia un bucle infinito que "colgaria" el programa, ya que ninguna instruccion modificaria b si llega alguna vez a valer 1 (esto podria ocurrir solo si se tuviesen hilos y se estuviera compartiendo la variable b, pero eso es harina de otro costal). Asi que aunque continue y break pueden usarse en los bucles, no es muy recomendable y siempre se puede realizar lo mismo con otro codigo mas elegante y estructurado.

Volviendo al switch, hay que destacar que si no se utiliza break al final de un case el flujo de programa continuara con el siguiente case. Un ejemplo puede ser el siguiente: queremos que si el numero es 1,2 o 3 la variable menor_que_cuatro tome el valor 1, pero si el numero es 1 tambien queremos que la variable es_uno tome el valor 1. En otro caso, las variables anteriores deberan valer 0. Y queremos hacerlo con switch. Esto seria:

```
        switch(numero){
        case 1:
                es_uno=1;
                /* Sigue con case 2... */
        case 2:
        case 3:
                menor_que_cuatro=1;
                break;
        default:
                es_uno=menor_que_cuatro=0;
                break;
        }
```

Vemos que case 2 y case 3 comparten el codigo. Tambien ocurre lo mismo con case 1, pero ademas hay una parte especifica (es_uno=1;) debido a como hemos

estructurado el switch sin break. Para evitar que en una revision posterior pensemos que hemos olvidado el break y lo añadamos por error, se suele indicar que el break no hace falta con un comentario (en el ejemplo, con el comentario /* Sigue con case 2... */).

- goto y etiquetas.

La palabra reservada goto permite desviar el flujo de programa a la posicion donde este situada la etiqueta a la que se refiere. Esta etiqueta puede declararse antes o despues del goto, el compilador sabra buscarla por todo el archivo correspondiente. La sintaxis es:

```
goto etiqueta;
.....
.....
etiqueta:
```

El uso de goto y etiquetas esta totalmente desaconsejado porque crean codigo ilegible, pero si pensais que pudiera haceros falta alguna vez, ahi esta. Como comentario, dire que yo nunca he utilizado un goto en mi codigo.

2.2 Funciones.

Hasta ahora hemos visto como se consigue hacer un bucle que repita cierta parte del codigo un numero determinado de veces o hasta que se cumpla una condicion, y como hacer que un bloque de codigo solo se ejecute si se da una condicion determinada.

Con estas herramientas podriamos hacer un programa real, pero seguramente habria partes del codigo que tendriamos que repetir, y todo estaria escrito como un continuo de codigo infumable. Las funciones nos proporcionan la forma de hacer nuestro codigo mas compacto, conciso, elegante y facil de leer. Ademas, las funciones permiten la reutilizacion de codigo y son la base de las bibliotecas (y en C++ los metodos, que no son mas que funciones, son una parte muy importante de la posibilidad de reutilizacion de codigo).

Una funcion es, en parte, muy parecida a un operador porque puede tomar parametros (algo asi como operandos) y porque puede devolver un valor. Pero tambien, puesto que puede ejecutarse las veces que haga falta escribiendola solo una vez, se parece a un bucle.

Una funcion no es mas que un subprograma o subrutina. Es decir, una seccion de codigo que puede ser invocada todas las veces que sea necesario. Por ejemplo, supongamos que tenemos un programa que va leyendo cada linea de un archivo de texto y busca en cada una de ellas la palabra "password". Podriamos un bucle que, mientras no se acabase el fichero, leyese una linea del programa cada vez y buscase en esa linea la palabra password. Pero tambien podriamos hacer dos funciones: una que leyese la siguiente linea del fichero y otra que buscase la palabra password en una linea de texto. Luego veremos algo asi.

- Declaracion.
Para escribir una funcion, debemos declararla y definirla. La declaracion es simplemente indicar el tipo de datos que devolvera la funcion, su nombre y los parametros que puede tomar. Vamos a escribir la declaracion de una funcion que se llame multiplica, que tome como parametros dos valores short y que devuelva un valor int:

Podemos indicar solo el tipo de los parametros o tambien el identificador de los mismos. Para indicar que el primer parametro se llamara numero1 y el segundo numero2 seria:

```
int multiplica (short numero1, short numero2);
```

El tipo del valor de retorno de la funcion puede obviarse, con lo que se tomara el tipo por defecto, int:

```
multiplica(short, short);
```

Tambien puede hacerse una funcion que no tome ningun parametro mediante la palabra clave void:

```
int multiplica(void);
```

E incluso podemos hacer una funcion que no devuelva ningun valor, lo que en otros lenguajes (i.e. Pascal) se conoce como un procedimiento:

```
void multiplica(short, short);
```

- Definicion.

La definicion de la funcion no es mas que "llenar" ese prototipo de codigo, escribir el codigo que la funcion va a ejecutar. La definicion de la funcion puede hacerse a la vez que la declaracion o mas tarde en el fichero fuente o en otro fichero, como ya veremos cuando veamos los ficheros de cabecera o headers. El codigo que compone la funcion se indica en la definicion de la misma, y se encierra entre {}.

Para definir la funcion multiplica() con declaracion:

```
int multiplica (short, short);
```

podemos hacer:

```
int multiplica(short n1, short n2){
        return n1*n2;
}
```

Vemos que en la definicion es OBLIGATORIO dar un identificador para cada parametro. El identificador no tiene por que coincidir con el identificador de la declaracion, si es que se les dio nombre. De cualquier modo, es recomendable que los identificadores coincidan para no liar el codigo.

Lo que si que debe coincidir en la declaracion y la definicion son tanto los tipos del valor de retorno como los de cada parametro. Tambien debe coincidir el numero de parametros que toma la funcion. No se permite la sobrecarga de funciones como en los lenguajes orientados a objetos (C++, Java, Object-Pascal, SmallTalk, etc.). Por tanto, no es valido:

```
int multiplica (short);
int multiplica (short a, short b){
        return a*b;
}
```

Intentar compilar un programa con ese codigo genera el siguiente error en el caso del compilador GCC:

```
program.c:4: conflicting types for ’multiplica’
program.c:3: previous declaration of ’multiplica’
```

La palabra reservada return termina con la ejecucion de la funcion e indica cual es el valor que debe devolver. En el ejemplo, vemos que la funcion devuelve el valor a*b. Por lo tanto, hemos hecho una funcion que toma dos valores short, los multiplica y devuelve el resultado como un int. No es muy util, ¿no? Bueno, esperad un poco.

- La funcion main.

Todo programa en C consta de la funcion principal o funcion main(). Esta funcion es la primera que se ejecuta y cuando termina el programa tambien termina. Es la funcion madre de todas las demas, puesto que cualquier otra funcion ha debido ser llamada desde la funcion main(). Por defecto toma el tipo int, aunque podemos indicar que devuelva otros tipo, i.e. void.

Generalmente el compilador emitira un "warning" si el tipo de main no es int. Si main toma tipo int, el valor de retorno de main puede servir para indicar cualquier condicion al acabar el programa (fin con exito, error de disco, etc.).

La funcion main puede tomar parametros, que se utilizan generalmente para procesar los argumentos en linea de comandos, pero tambien podemos "pasar de ellos" indicando que su unico parametro es void, o sencillamente escribiendo:

```
        main(){
        ....
        }
```

Finalmente, hemos de decir que la funcion main no debe declararse. Solo la definicion es necesaria.

Podemos entonces escribir nuestro primer programa compilable en C. Para ello, creamos un fichero de texto ascii de nombre primero.c con cualquier editor de textos (preferiblemente Joe’s Own Editor :D) o con el editor incorporado en la herramientas de programacion si utilizais entornos de compilacion de tipo Borland Builder, etc. El contenido de dicho fichero debe ser:

```
int main(void){
        int a=1;
        int b=2;
        int c;

        c = a*b;
        return 0;       /* Un valor de retorno 0 indica exito */
}
```

Podemos compilar el programa. Con el compilador GCC la orden es:

```
        gcc fichero.c
```

Ya podemos ejecutarlo. En linux sera:

```
        ./a.out
```

Si hacemos esto no veremos nada nuevo, simplemente de nuevo el indicador del shell "khorrorsive$ ", "C:\> " en Windos/MS-DOS. Pero el programa se habra ejecutado, habra reservado espacio para tres variables, habra iniciado dos de ellas a 1 y 2 respectivamente, las habra multiplicado y habra almacenado el resultado en la tercera variable y finalmente habra terminado el programa con un valor de retorno 0.

Vale, esto puede parecer una mierda de categoria. Quereis ver algo en la pantalla que os muestre que somos la hostia programando y que podemos hacer un m3g4-h4x0r-programa que multiplique dos constantes ¿no? :-P Bueeeeeeeno. Entonces vamos a añadir un poco de "magia" que comprendereis mas adelante. El programa ahora debe ser:

```
#include <stdio.h>

int main(void){
        int a=1;
        int b=2;
        int c;

        c = a*b;
        printf ("%d x %d = %d\n",a,b,c);
        return 0;       /* Un valor de retorno 0 indica exito */
}
```

```
Lo compilamos, lo ejecutamos, y.... ¡Tachan! la salida es:
khorrorsive$ ./a.out
1 x 2 = 2
khorrorsive$

Muy bien. Ahora podeis cambiar los valores a los que iniciamos a y b y
recompilar para ver que realmente multiplica estos dos valores. O podemos
hacer algo un poco mas complicado, como utilizar una funcion multiplica()
similar a la que hicimos anteriormente. Para ello:

#include <stdio.h>

int multiplica (int n1, int n2){
        return n1*n2;
}

int main(void){
        int a,b,c;
        a=1;
        b=2;
        c=multiplica(a,b);

        printf("%d x %d = %d\n",a,b,c);
        return 0;
}

Si compilamos y ejecutamos este nuevo programa, su salida es de nuevo:
khorrorsive$ ./a.out
1 x 2 = 2
khorrorsive$

Pero ahora hemos utilizado una llamada a nuestra funcion multiplica(). La
parte multiplica(a,b) es la llamada a funcion.

- Explicacion exhaustiva del ejemplo.

Puede que esto de las funciones te confunda. Voy a intentar explicarlo
mejor. Para ello tomare el ultimo programa y lo explicare paso a paso:

        #include <stdio.h>
Esta primera linea no es mas que una directiva para el preprocesador que le
dice que antes de comenzar la compilacion busque el fichero stdio.h en los
directorios "include" por defecto (el compilador esta configurado y sabe
donde tiene que buscar) y que el compilador lo procese antes que el resto del
fichero. Esto se hace porque la funcion printf() que se utiliza mas tarde es
una funcion de la biblioteca estandar cuya DECLARACION esta escrita en este
fichero. Es decir, el include se utiliza para que el compilador sepa que
existe la funcion printf() y sepa su prototipo (parametros, tipo del valor de
retorno, etc.).

        int multiplica (int n1, int n2){
                return n1*n2;
        }
Esta parte ya hemos visto que es la declaracion y la definicion de una
funcion, de nombre multiplica, que toma dos parametros de tipo int (n1 y
n2) y que devuelve un valor de tipo int. Asimismo, escribimos el codigo
correspondiente a la funcion, que consiste solo en que devuelva el resultado
de multiplicar ambos parametros.

        int main(void){
                int a,b,c;
```

```
                a=1;
                b=2;
```

Esta parte comienza la definicion de la funcion main(), indicando primero que devuelve un valor de tipo int y que no toma parametros. Luego declara tres variables de tipo int a,b y c, y almacena en las dos primeras los valores 1 y 2 respectivamente.

```
                c=multiplica(a,b);
```

Esto llama a la funcion multiplica(), pasandole como parametros los valores almacenados en a y b. El valor de retorno de dicha funcion lo almacena en la variable c. Puesto que la funcion multiplica() lo que devuelve es el resultado de multiplicar los parametros de entrada, c almacena ahora el resultado de multiplicar a por b.

```
                printf("%d x %d = %d\n",a,b,c);
```

Esta es quiza la parte mas complicada. Lo unico que hace es invocar a la funcion printf, pasandole como parametros la constante de cadena "%d x %d = %d\n", el valor almacenado en a, el valor almacenado en b y el valor almacenado en c.

printf() pertenece a un tipo especial de funciones que pueden tomar una lista variable de parametros. En el caso de printf(), el primer parametro debe ser una "cadena de formato" (en realidad, lo que le pasamos es un PUNTERO al primer caracter de la cadena) que indica que es lo que hay que mostrar por pantalla. Cada vez que aparezca el patron %d le estaremos indicando a printf que debe escribir un numero en decimal (base 10). En nuestro caso, le decimos que escriba un numero en base 10, el caracter ’ ’ (espacio), el caracter ’x’, el caracter ’ ’, otro numero en base 10, el caracter ’ ’, el caracter ’=’, el caracter ’ ’ y otro numero en base 10. Y ¿que numeros seran los que escriba?. Pues el primero de ellos sera el siguiente parametro (a), el segundo el siguiente (b) y el tercero el ultimo (c). Entonces la salida de printf sera "1 x 2 = 2". Es decir, a x b = c.

```
                return 0;
```

Finalmente, hacemos main() termine devolviendo un valor 0.

- Funciones anidadas.

Bien, espero que la cosa este algo mas clara, porque ahora vamos dar otra vuelta de tuerca. Vamos a escribir el programa anterior, pero eliminando la variable c.

Si nos fijamos, la variable c solo nos sirve para almacenar el valor multiplica(a,b). Y ese valor solo lo utilizamos una vez, para pasarlo a printf() y que lo saque por pantalla. Asi que lo que vamos a hacer es poner la llamada a multiplica como el ultimo parametro de printf(), en la posicion anterior de c:

```
#include <stdio.h>

int multiplica (int n1, int n2){
        return n1*n2;
}

int main(void){
        int a,b;
        a=1;
        b=2;

        printf("%d x %d = %d\n",a,b,multiplica(a,b););
        return 0;
}
```

En este caso, el compilador ve que el ultimo parametro de printf() es precisamente el valor de retorno de multiplica(). Por ello el programa primero ejecuta la funcion multiplica() pasandole como parametros los valores almacenados en a y b, y luego llama a printf() pasandole como parametros la cadena de formato, a, b, y el valor devuelto por multiplica(). Las llamadas a funciones pueden estar anidadas, como estamos viendo. De hecho, es un caso muy comun y utilizado, no es nada oscuro del lenguaje.

- Parametros por valor y por referencia.

NOTA: Para comprender esta seccion es necesario saber sobre punteros en C. Asi que si aun no has leido la seccion de punteros, simplemente pasa esta y vuelve cuando sepas utilizar punteros.

En programacion, los parametros pueden pasarse por valor o por referencia. Si se pasan por valor, lo que se esta pasando en realidad es una COPIA del parametro. La funcion puede modificar el parametro a su antojo, pero solo estara modificando su copia interna. Durante toda la ejecucion de la funcion y una vez esta termine, el valor original que se copio sigue valiendo lo mismo y no se ha visto alterado.

Cuando se pasa un parametro por referencia, no se pasa el parametro en si, sino la direccion de memoria donde esta almacenado ese valor. De forma que ahora solo hay una copia de ese valor que se comparte entre la funcion padre y la funcion hija que ha sido llamada. Y si la funcion hija modifica ese parametro, lo modifica tambien para la funcion padre.

Todos los parametros en C se pasan por valor. C no permite el paso de parametros por referencia, como ocurre en Pascal. Para conseguir un paso por referencia, lo que se hace en realidad es pasar por valor un puntero al parametro. De esta forma, la funcion hija tiene una copia de la direccion de memoria donde se almacena el valor con el que tiene que trabajar. Hay una diferencia sutil con el paso de parametros por referencia de Pascal: aqui el programador *sabe* en cada momento que esta trabajando con un puntero al valor deseado, mientras que en Pascal la sintaxis es la misma salvo en la declaracion del tipo del parametro.

Por ejemplo podemos hacer en C:

```
void cambia(int *pa){
        *pa = 1;
}

int main(void){
        int a = 0;
        cambia(&a);
        printf("a vale: %d\n",a);
        return 0;
}
```

Al ejecutar el programa anterior, la salida sera:

```
khorrorsive$ ./a.out
a vale 1
khorrorsive$
```

Pero si en vez de utilizar el puntero hacemos:

```
void cambia(int pa){
        pa = 1;
}

int main(void){
        int a = 0;
```

```
        cambia(a);
        printf("a vale: %d\n",a);
        return 0;
}
```

```
La salida sera:
khorrorsive$ ./a.out
a vale 0
khorrorsive$
```

Vemos que en este caso la variable a no ha cambiado, aunque dentro de la funcion cambia() hayamos almacenado un valor 1 en el parametro.

- Un ejemplo de parametros "por referencia".

Bien, vamos a aprovechar esto que ya sabemos para escribir un programa que multiplique dos numeros algo mas elaborado que el anterior. Esta vez no voy a escribir primero el programa y luego lo explico, sino que lo vamos a hacer al reves: primero lo pensamos, y luego lo escribimos.

Esta vez vamos a hacer un programa que tome los numeros a multiplicar desde el teclado, de forma que no haya que retocar el programa y recompilarlo cada vez que queremos multiplicar numeros distintos. Y para ello vamos a utilizar la funcion scanf().

Al igual que printf(), scanf() es una funcion de la biblioteca de entrada salida estandar, por lo que debemos dar a conocer su prototipo al compilador mediante "#include <stdio.h>". De forma similar a lo que pasaba con printf(), scanf() recibe un numero variable de parametros, el primero de los cuales es un puntero a una cadena de caracteres de formato. En esta cadena de formato le indicaremos a scanf() que es lo que queremos que reciba por teclado. En nuestro caso, queremos que scanf reciba dos numero enteros, por lo que nuestra cadena de formato sera "%d %d" (traduciendo, dos numeros enteros en forma decimal, base 10 para los amigos).
Los siguientes parametros deben ser las variables donde queremos almacenar estos valores... ¡Un momento! ¿Las variables? ¡Noooo! Si pasaramos las variables, en realidad estariamos pasando a scanf() UNA COPIA de sus valores. Y lo que queremos es que scanf() *modifique* estos valores. Es decir, queremos pasar las variables "por referencia", de forma que scanf() almacene en ellas los numeros enteros que el usuario introduzca por teclado. Por lo tanto, lo que debemos pasarle a scanf() son PUNTEROS a las variables. Para ello usaremos el operador & (direccion de).

Una vez scanf() haya recibido y procesado los valores, las variables tendran almacenados los mismos. Solo nos queda llamar a printf() para escribir el resultado, al igual que pasaba en el ejemplo anterior. De nuevo, vamos a decirle a printf() que escriba "Entero1 x Entero2 = Resultado", es decir, nuestra cadena de formato para printf() sera "%d x %d = %d\n" (no olvideis incluir el caracter de nueva linea al final, para que quede mas bonito), y el resto de argumentos debe ser las dos variables y la multiplicacion de ambas, respectivamente. ¿Por que no hay que pasarle punteros a las variables a printf()? Porque printf() *no va a modificarlas*, solo va a escribir su valor por pantalla.

Entonces, nuestro programa seria:

```
#include <stdio.h>

int main(void){
        int n1, n2;
        scanf("%d %d", &n1, &n2);       /* Recibe los numeros */
        printf("%d x %d = %d\n",n1,n2,n1*n2);   /* Imprime la operacion */
```

```
        return 0;       /* Indicamos exito en la ejecucion del programa */
}
```

Si compilais y ejecutais dicho codigo, vereis que el programa se queda a la espera de que introduzcais dos numeros. Si lo haceis, imprimira el resultado. Tened en cuenta que si en vez de dos numeros introducis una cadena de caracteres o cualquier otra cosa, el programa no funcionara. Esto es asi porque no he incluido control de errores en la entrada, para no complicar la cosa. Podeis probar por curiosidad a introducir cualquier chorrada no numerica a ver que sale.

- Argumentos en linea de comandos.

Otro caso util que podemos ver para aprender a utilizar funciones, argumentos y punteros es el del tipico programa que recibe argumentos en linea de comandos.

Aunque los sistemas con interfaz grafica que Windows ha puesto de moda han conseguido que haya usuarios que nunca hayan escrito un comando en el interprete de comandos, vamos a arriesgarnos a pulsar el boton de "Instalacion Personalizada (solo para usuarios avanzados)" y a aprender algo util.

Los programas pueden recibir argumentos en linea de comandos. Por ejemplo, el caso mas sencillo es quizas el comando "cd" que tanto en Unix como en M$-DoS/Windows sirve para cambiar el directorio (¿o ahora se llaman carpetas? :P) activo. El comando "cd" podria no ser mas que un programa al que se le pasase, en linea de comandos, el nombre del directorio al que queremos cambiar. Muy requetebien. A ver como cohones se programaria eso.

Sencillo, hombre, no empieces a sudar. Para ello, vamos a ampliar un poco la definicion de nuestra funcion main(). Ahora, la funcion main() va a tomar dos parametros: un int -que para seguir una tradicion ancestral llamaremos argc- y un char ** -llamado argv-. Entonces...

P: Espera, espera, que ahi habia demasiados *. ¿Que leche de tipo tiene argv?
R: argv no es mas que un puntero a puntero a caracter.
P: ¿Como se come eso? Meloxpliquen.
R: Facil hombre, miralo mejor como si fuera char* argv[].
P: Ahhhh, eso es mas facil ¿no? Una tabla de punteros a caracter, es decir, una tabla de cadenas de caracteres ¿Verdad?
R: Ahi l'as dao.
P: Vale, ¿y para que sirve cada uno de los parametros?
R: argc es facil, hombre. argc indica cuantos parametros (incluido el nombre del programa) ha escrito el usuario.
P: Entonces si yo llamo a mi programa con la orden "./a.out", argc valdra 0, ¿no?
R: ¿No te estoy diciendo que "incluido el nombre de programa"? Si escribes la orden "./a.out", argc valdra 1.
P: Vale, vale, no te mosquees. ¿Y argv?
R: argv es una tabla, de argc elementos, cada uno de los cuales es una cadena de caracteres. Entonces la primera cadena es el nombre del programa, la segunda el primer argumento, etc.
P: ¡Ah, estupendo! Entonces, ¿el nombre del programa es argv[1]?
R: No, ceporro. Las tablas en C comienzan por el indice 0. argv[0] es el nombre del programa, y argv[1] es el primer argumento (si existe).
P: Vale, pero todos mis programas se llaman a.out (trabajo en Unix). ¿Para que voy a usar entonces argv[0]?
R: Si no quieres que tu programa se llame a.out, utiliza la opcion -o nombre_del_programa con el compilador de C. Por ejemplo:

```
gcc programa.c -o miprograma
```

P: Y entonces, ¿argv cuantas filas tiene?

R: Pues depende de cada llamada. Precisamente argv no se sabe cuantas filas tiene hasta que el programa no se esta ejecutando. Es un claro ejemplo datos cuyo tamaño no es conocido en tiempo de compilacion. Por eso se utiliza argc, para que el programador pueda saber cuantos argumentos se han pasado en linea de comandos en cada ejecucion.
P: Entiendo. Bueno, la ultima pregunta. ¿Por que los llamas argc y argv?
R: Utilizo esos nombres porque son los que usa todo el mundo, aunque puedes ponerle los nombres que quieras. argc significa ARGument Count, y argv significa ARGument Vector, nombres que vienen a describir bastante bien el contenido y uso de dichas variables.

Bien, despues de este dialogo de besugos, ya sabemos como hay que utilizar argc y argv. Vamos a empezar por un ejemplo sencillo: hacer un programa que no acepte ningun parametro. Lo voy a hacer con el operador ternario para que penseis un poco y os acostumbreis a ello:

```
#include<stdio.h>
int main(int argc, char *argv[]){
        printf(argc==1?"OK\n":"¡Error! No admito argumentos en linea\n");
        return argc==1?0:-1;
}
```

Vale. Ya esta. Ahora vamos a hacer un programa que escriba la linea de comandos al completo, comenzando por el nombre del programa y siguiendo con todos los argumentos en linea que le hayamos pasado. Para ello, contare algo mas sobre la cadena de formato de printf().
Si queremos que printf() escriba una cadena de caracteres que no sea constante, podemos utilizar la secuencia %s. Es decir, la proposicion

```
        printf("%s\n",cadena);
```

escribira en pantalla el contenido de la cadena de caracteres "cadena" hasta que encuentre el terminador, y luego el caracter ’\n’ (nueva linea). Entonces nuestro programa debe ser:

```
#include <stdio.h>
int main(int argc, char *argv[]){
        int i=0;
        while(argc--){
                printf("%s ",argv[i++]);
        }
        printf("\n");
        return 0;
}
```

Puede que todo esto no te parezca muy util. ¿Para que sirve un programa que escribe los argumentos que recibe? Bueno, para nada; es solo un ejemplo. Pero ya sabes como consigue el compilador gcc averiguar que has utilizado la opcion -o nombre_del_programa para que al compilar tu programa no se llame a.out. Y, lo mas importante, ya puedes ir corriendo a contarselo a tu padre/amigo/hermano/novia para que te mire con cara de "me la suda, colgao" y te diga: "Me da igual, me voy a ver O.T./G.H./Aqui hay tomate/Er furbol" :-/

EOF

```
-[ 0x07 ]-----------------------------------------------------------------
-[ Proyectos, Peticiones, Avisos ]-----------------------------------------
-[ by SET Ezine ]----------------------------------------------------SET-29--
```

Si, sabemos es que esta seccion es muyyy repetitiva (hasta repetimos este parrafo!), y que siempre decimos lo mismo, pero hay cosas que siempre teneis que tener en cuenta, por eso esta seccion de proyectos, peticiones, avisos y demas galimatias.

Como siempre os comentaremos varias cosas:
- Como colaborar en este ezine
- Nuestros articulos mas buscados
- Como escribir
- Nuestros mirrors
- En nuestro proximo numero
- Otros avisos

-[Como colaborar en este ezine]----------

Si aun no te hemos convencido de que escribas en SET esperamos que lo hagas solo para que no te sigamos dando la paliza, ya sabes que puedes colaborar en multitud de tareas como por ejemplo haciendo mirrors de SET, graficos, enviando donativos (metalico/embutido/tangas de tu novia (limpios!!!)) tambien ahora aceptamos sujetadores pero en ningun caso inferiores a la talla 80 ni de primera mano, sorprendenos!

-[Nuestros articulos mas buscados]----------

Articulos, articulos, conocimientos, datos!, comparte tus conocimientos con nosotros y nuestros lectores, buscamos articulos tecnicos, de opinion, serios, de humor, ... en realidad lo queremos todo y especialmente si es brillante. Tampoco es que tengas que deslumbrar a tu novia, que en ningun momento va a perder su tiempo en leernos, pero si tienes la mas minima idea o desvario de cualquier tipo, no te quedes pensando voy a hacerlo... hazlo!.

Tampoco queremos que te auto-juzges, deja que seamos nosotros los que digamos si es interesante o no.
Deja de perder el tiempo mirando el monitor como un memo y ponte a escribir YA!.

Como de costumbre las colaboraciones las enviais indistintamente aqui:
<set-fw@bigfoot.com>
<web@set-ezine.org>

Para que te hagas una idea, esto es lo que buscamos para nuestros proximos numeros... y ten claro que estamos abiertos a ideas nuevas....

- articulos legales: ya tenemos uno de la LSSI, ¡nos faltan derechos de autor! nadie sabe nada?

- sistemas operativos: hace tiempo que nadie destripa un sistema operativo en toda regla ¿alguien tiene a mano un AS400 o un Sperry Plus?

- Retro informatica. Has descubierto como entrar en la NASA con tu Spectrum 48+? somos todo ojos, y si no siempre puedes destripar el SO como curiosidad

- Programacion: cualquier lenguaje interesante, guias de inicio, o de seguimiento, no importa demasiado si el lenguaje es COBOL, ADA, RPG, Pascal, no importa si esta desfasado o si es lo ultimo de lo ultimo, lo importante es que se publique para que la informacion este a mano de todos, eso si, No hagais todos manuales de C!!!

- Chapuzing electronico: Has fabricado un aparato domotico para controlar la temperatura del piso de tu vecina? estamos interesados en saber como lo has hecho...

- Domotica: No importa si es software o hardware... informacion!!!!

- Evaluacion de software de seguridad: os veo vagos, Nadie le busca las cosquillas a este software?

- Hacking, craking, virus, preaking, sobre todo cracking!

- SAP.. somos los unicos que gustan este juguete? Me parece que no, ya que hemos encontrado a alguien con conocimientos, pero: alguien da mas?

- ORACLE, MySQL, MsSQL, postgrees.. Aqui tambien nos hemos topado con un entendido en la materia, pero la union hace la fuerza. Alguien levanta el dedo ?

- Mariconeos con LOTUS, nos encanta jugar con software para empresas, un gran olvidado del hacking "a lo bestia".

- Vuestras cronicas de puteo a usuarios desde vuestro puesto de admin...

- Usabilidad del software (acaso no es interesante el tema?, porque el software es tan incomodo?)

- Wireless. Otro tema que nos encanta. Los aeropuertos y las estaciones de tren en algunos paises europeos nos ofrecen amplias posibilidades de curiosear en lo que navega sobre las ondas magneticas. Nadie se ha dedicado a utilizar las horas tontas esperando un avion en rastrear el trafico wireless ?

- Redes libres. Alguien esta haciendo un seguimiento de Freenet ? A que se debe el bajo rendimiento que padece ultimamente ?

- Juegos on-line. Todo un mundo que, egocentricamente, no da senyales de vida en nuestra paginas electronicas

- Lo que tu quieras...

Tardaremos en publicarlo, puede que no te respondamos a la primera (si, ahora siempre contestamos a la primera y rapido) pero deberias confiar viendo nuestra historia que SET saldra y que tu articulo vera la luz en unos pocos meses, salvo excepciones que las ha habido.

-[Como escribir]----------

Esperemos que no tengamos como explicar como se escribe, pero para que os podais guiar de unas pautas y normas de estilo (que por cierto, nadie cumple y nos vamos a poner serios con el tema), os exponemos aqui algunas cosillas a tener en cuenta.

SOBRE ESTILO EN EL TEXTO:

- No insulteis y tratar de no ofender a nadie, ya sabeis que a la minima salta la liebre, y SET paga los platos rotos

- Cuando vertais una opinion personal, sujeta a vuestra percepcion de las cosas, tratar de decir que es vuestra opinion, puede que no todo el mundo opine como vosotros, igual nisiquiera nosotros.

- No tenemos ni queremos normas a la hora de escribir, si te gusta mezclar tu articulo con bromas hazlo, si prefieres ser serio en vez de jocoso... adelante, Pero ten claro que SET tiene algunos gustos muy definidos: ¡Nos gusta el humor!, Mezcla tus articulos con bromas o comementarios, porque la verdad, para hacer una documentacion seria ya hay mucha gente en Internet.
  Ah!!!!, no llamar a las cosas correctamente, insultar gratuitamente a empresas, programas o personas NO ES HUMOR.

- Otra de las cosas que en SET nos gusta, es llamar las cosas por su nombre, por ejemplo, Microsoft se llama Microsoft, no mierdasoft, Microchof o cosas similares, deformar el nombre de las empresas quita mucho valor a los articulos, puesto que parecen hechos con prejuicios.

SOBRE NORMAS DE ESTILO

- Tratad de respetar nuestras normas de estilo!. Son simples y nos facilitan mucho las tareas. Si los articulos los escribis pensando en estas reglas, sera mas facil tener lista antes SET y vuestro articulo tambien alcanzara antes al publico.

- 80 COLUMNAS (ni mas ni menos, bueno menos si.)

- Usa los 127 caracteres ASCII, esto ayuda a que se vea como dios manda en todas las maquinas sean del tipo que sean. El hecho de escribirlo con el Edit de DOS no hace tu texto 100% compatible pero casi. Mucho cuidado con los disenyos en ascii que luego no se ven bien. Sobre las enyes (¤).

- Y como es natural, las faltas de ortografia bajan nota, medio punto por falta y las gordas uno entero.

  Ya tenemos bastante con corregir nuestras propias faltas.

- Ahorraros el ASCII ART, porque corre serio riesgo de ser eliminado.

- Por dios, no utilizeis los tabuladores, esta comprobado que nos levantan un fuerte dolor de cabeza cuando estamos maquetando este E-zine.

-[Nuestros mirrors]----------

http://salteadores.tsx.org - USA

http://www.hackemate.com.ar/ezines/set/ - Argentina

El resto que nos aviso de tener un mirror, o no lo encontramos o las paginas estaban desactivadas, ¡mala suerte!.

Existe una posiblidad, la mas posible de todas y es el extravio de correo, que nos sucede mas amenudo de lo que debieramos....

-[En nuestro proximo numero]----------

Antes de que colapseis el buzon de correo preguntando cuando saldra SET 30 os respondo: Depende de ti y de tus colaboraciones.

En absoluto conocemos la fecha de salida del proximo numero, pero en un esfuerzo por fijarnos una fecha objetivo pondremos..... septiembre de 2004

-[Otros avisos]----------

Este es un pequenyo aviso para las pocas personas que nos envian material
'fisico': CD's, revistas y cosas de esas.... (por cierto, todavia no hemos
recibido ningun paquete envuelto en billetes de 100 euros... a que esperais?)

El caso es que ya no disponemos de nuestro tradicional apartado de correos
o sea que todo aquel que nos quiera enviar algo 'fisico' que se ponga en
contacto previamente con nosotros por e-mail, O sea que si ardeis en deseos
de enviarnos el video de vuestro ultimo revolcon con vuestra novia, avisanos
antes.

(no me cansare de repetir las cuentas de correo)

<web@set-ezine.org>
<set-fw@bigfoot.com>

EOF

```
-[ 0x08 ]-------------------------------------------------------------------
-[ VMS ]--------------------------------------------------------------------
-[ by tmp-paseante ]-----------------------------------------------SET-29--
```

Todo lo aqui escrito es puramente informativo, Cualquier uso que hagais de lo
aqui escrito es bajo vuestra propia responsabilidad.
Aunque la verdad, no se que mas podeis hacer con ello.

****

VMS el sistema operativo que Windows NT 8.0 desearia ser**
(Publicado en la pagina web de OpenVMS por un par de horas)

INTRODUCCION

VMS son las siglas para Sistema de Memoria Virtual. Fue desarrollado por
Digital (actualmente Compaq, actualmente HP) hace 25 anhos cuando Digital
paso de fabricar PDPs a fabricar la VAX. Que para que quieres aprender esa
antigualla, bueno se supone que un hacker le gusta la informacion por si misma
no por su utilidad, aparte de que aunque no muy publicamente, aun sige siendo
ampliamente utilizado en sitios sin importancia tales como bases de datos
medicos, bancos (el Credit Lyonesse en Paris por ejemplo), loterias, el backbone
de el exchange de la bolsa de Nueva York o el de la nueva bolsa electronica de
Chicago.. (Lo que yo decia sitios sin importancia:).
Que por que lo usan en esa clase de sitios, bueno eso es facil, Clusters.

Que es eso?
Cuando conectas varios ordenadores y los haces ver como uno para un usuario que
se conecte al sistema.

Que si que has oido hablar de eso, que Windows NT y los unix lo tienen.

Si claro y Windows95 es multiusiario no te fastidia.

Mejor lo dejamos para luego, pero solo dire que:
Los Cluster de VMS es el baremo al que aspiran todos los demas clusters.

Que si es tan cojonudo por que no oyes hablar mas de el.
Las dos principales razones:

- Esta mas en liga con sistemas operativos tipo mainframe de IBM que con otra
  cosa, tampoco se oye hablar mucho de ellos.
- La publicidad de Digital ha sido pesima (por que crees sino que la compro
  Compaq que luego se junto con HP, buenos productos pero un marketing penoso).

Ahora que ya os he aburrido bastante, y me he quedado solo con los que
realmente tienen algun interes. Vamos a empezar a dar algo de informacion

Primero vamos alla con el lio de los nombres.
Como ya dije este sistema operativo se desarrollo para las VAX, Es mas, es
posible que en el centro de calculo de tu universidad tengan algun sistema VMS
que es conocido como "LA VAX" aunque haga anhos que ya no es una VAX.
Con posterioridad se le anhade un paquete que le hace cumplir con el extandar
POSIX y se cambia el nombre a OpenVMS, asi mismo se porta a procesadores Alpha
con lo que acabamos teniendo OpenVMS/VAX y OpenVMS Alpha.
Osea que OpenVMS y VMS son lo mismo, solo dos nombres para la misma cosa.
Ahora a por lo que te interesa el maldito sistema.

PRIMER ENCUENTRO

Haces telnet a una maquina y te encuentras con algo asi:

```
 Welcome to OpenVMS (TM) XXX Operating System, Version V%.%

Username:
```

Donde XXX puede ser VAX o ALPHA o AXP, (los dos ultimos significan que son alpha), y el %.% es la version.
Este es el mensaje que viene de fabrica en la maquina pero puede ser cualquier cosa que decida ponerle el administrador, lo que no cambia es el

```
Username:
```

Ahora las buenas noticias el sistema no distinge entre mayusculas y minusculas y las malas no se puede hacer mas de 3 intentos consecutivos, hay que rehacer la conexion y la longitud minima del pasword es 6 letras aunque no es raro que haya sido aumentado a nueve.

Y la peor de todas no importa lo inutil que sea el administrador cada vez que no consigas entrar queda registrado, es la configuracion por defecto.

Y la configuracion por defecto dice que si durante 24 horas un usuario realiza 9 intentos fallidos queda registrado como intento de intrusion y tres fallos mas y el usuario queda bloqueado del acceso por 24 horas desde el ultimo intento. Naturalmente sin decirte nada asi que no importa que apartir de ese momento des el password correcto, el ordenador te negara el acceso como si fuera incorrecto, solo retrasara el tiempo que tardara en poder entrar. Eso o hasta que hable con el administrador y le limpie el la alerta de seguridad.

Hey ! pero eres un super hacker no?, asi que ya sabes coges tu linea directa con dios y le pides un nombre de usuario y un pasword y ya esta.

Acabas de conseguir entrar.

Y ahora que?

El sistema esta basado en ingles es decir los comandos propios del sistema operativo son lo que quieres hacer dicho en ingles.

Pongamos un ejemplo sencillo. Digamos que te acabas de conectar con un telnet y las cosas no se ven como tu esperas vamos que la terminal no chuta bien (caso normal si usas la hyperteminal del windoze). Asi que quieres que el sistema le vuelva a preguntar las propiedades a la terminal pues.

```
SET TERMINAL/INQUIRE
```

O lo que es lo mismo en castellano

Pon la terminal preguntando.

Que eso es escribir demasiado, ohh no es gran problema, solo necesitas poner el numero de letras necesario para que no haya dos comandos que tengan las mismas letras 3 o 4 a lo mas es suficiente

```
SET TER/INQ
```

Asi que con un ligero conocimiento de ingles y el comando magico uno se puede convertir en un usuario como si tal cosa.
Que cual es el comando magico muy sencillo ?

```
HELP (ayuda)
```

Que te da el resultado siguiente

```
HELP
```

```
   The   HELP   command   invokes   the  HELP  Facility  to  display
   information about a command or topic.  In response to the "Topic?"
   prompt, you can:

      o Type  the  name of the command  or topic for which you need help.

      o Type  INSTRUCTIONS  for more detailed  instructions on how to use
        HELP.

      o Type  HINTS  if you are not  sure of the  name  of the command or
        topic for which you need help.

      o Type HELP/MESSAGE for help with the HELP/MESSAGE command.

      o Type a question mark (?) to redisplay the most recently requested
        text.



Press RETURN to continue ...

      o Press the Return key one or more times to exit from HELP.

   You  can  abbreviate any  topic name, although ambiguous abbreviations
   result in all matches being displayed.


  Additional information available:

  :=          =           @           ACCOUNTING ADVISE     ALLOCATE   ANALYZE
  APPEND      ASSIGN      ATTACH      AUTHORIZE  AUTOGEN    BACKUP     CALL
  CANCEL      CC          CLOSE       CMS        CONNECT    CONSOLE    CONTINUE
  CONVERT     COPY        CREATE      CXX        CXXDEMANGLE           CXXL
  CXXLINK     DCL_Tips    DEALLOCATE  DEASSIGN   DEBUG      DECK
  DECnet-Plus             DECset      DECthreads DEFINE     DEFRAGMENT DELETE
  DEPOSIT     DIAGNOSE    DIFFERENCES            DIRECTORY  DISABLE
  DISCONNECT  DISMOUNT    DPML        DSR        DSTGRAPH   DTM        DUMP
  DXML        EDIT        ENABLE      ENDSUBROUTINE         EOD        EOJ
  Errors      EXAMINE     EXCHANGE    EXIT       F90        FDL        FLOWGRAPH
  FONT        FORTRAN     FTP         GENCAT     GKS        GOSUB      GOTO
  HELP        Hints       ICONV       IF         INITIALIZE INQUIRE    INSTALL


Press RETURN to continue ...

  Instructions            JOB         KAP         LANCP      LATCP      Lexicals
  LIBRARY     LICENSE     Line_editing            LINK       LMCP       LOCALE
  LOGIN       LOGOUT      LPQ         LPRM        MACRO      MAIL       MERGE
  MESSAGE     MMK         MMS         MONITOR     MOUNT      Multimedia NCP
  NCS         NFS         ON          OPEN        PASSWORD   PCA        PHIGS
  PHONE       PIPE        POSIX       PPPD        PRINT      PRINT_Parameter
  PRODUCT     PSWRAP      PURGE       Queues      RCP        READ       RECALL
  RECOVER     RENAME      REPLY       REQUEST     RETURN     REXEC      RLOGIN
  RMS         RPCGEN      RSH         RTL_Routines            RUN        RUNOFF
  SCA         SCHEDULE    SEARCH      SET         SHOW       SMTP
  SOFTWINDOWS             SORT        SPAWN       START      STOP       SUBMIT
  SUBROUTINE Symbol_Assign            SYNCHRONIZE            SYSGEN     SYSMAN
  System_Services         Sys_Files   TCPIPTRACE  TELNET     TFF        TN3270
  TYPE        UCX         UNLOCK      V73_Features           VEST       VIEW
  VMSTAR      WAIT        WRITE
```

```
  Additional help libraries available (type @name for topics):

<aqui va otra lista de utilidades que el administrador haya querido anhadir>

Topic?

teclea el comando sobre el que quieres saber mas en topic y te dara mas
informacion de la que probablemente quieras saber.

Te aseguro que resulta un sistema muy facil de seguir.
Aun asi una forma de encontrar lo que andas buscando si no quieres nada
demasiado especial es

HELP HINTS

Que te dara como resultado.
HINTS

  Type the name of one of the categories listed below to obtain a list
  of related commands and topics.  To obtain detailed information on a
  topic, press the RETURN key until you reach the "Topic?" prompt and then
  type the name of the topic.

  Topics that appear in all uppercase are DCL commands.


  Additional information available:

  Batch_and_print_jobs  Command_procedures    Contacting_people
  Creating_processes    Developing_programs   Executing_programs
  Files_and_directories Logical_names         Operators_in_expressions
  Physical_devices      Security   System_management     Terminal_environment
  User_environment

HINTS Subtopic?


Ah.. estoy escribiendo los comandos en mayusculas pero es mayusculas insensible
o lo que es lo mismo no importa si escribes en mayusculas o minuscula o una
mezcla de todo ello. A el le da lo mismo.

O pero tu quieres ir sobre seguro quieres saber donde te estas metiendo antes
de arriesgarte que el administrador se de cuenta de que no eres quien se supone
que eres.

De acuerdo sales del sistema y empiezas a buscar informacion sobre este trasto.

La cosa es dificil hasta que encuentras

http://www.openvms.compaq.com/

<SET comment, puede que os encontreis mas a gusto en>
http://h71000.www7.hp.com/

y especialmente

http://www.openvms.compaq.com:8000/
o
http://www.openvms.compaq.com/doc/

(atentos a posibles cambios de direccion por la adquisicion por HP)
```

<SET,>
Probad en http://h71000.www7.hp.com/doc/

Donde se encuentra la documentacion disponible de VMS mas conocida como el muro, si has oido bien el muro, se le da ese nombre porque es aproximadamente lo que ocupa la version en papel un muro de estanterias lleno. En VMS esta documentado todo tu principal problema no es descubrir las cosas sino simplemente encontrar el manual correcto

Te has estado leyendo los manuales un poco por encima, la FAQ, y le has echado un vistazo a la lista de preguntas a los magos de esa misma web asi como un rapido vistazo a las webs listadas en la FAQ.

Veamos algunos puntos interesantes.

- El sistema se puede conseguir gratis (Hobby license) siempre que encuentres una maquina que lo pueda correr, Las Alphastation 255 usadas son baratas (relativamente y no son faciles de conseguir en Espanha), potentes para su edad (tienen 6-7 anhos) y en el peor caso corren Linux o bien instalas un simulador de VAX (simh y ts10 son gratuitos), pero te piden que te registres en DECUS que tambien es gratis pero, eso de estar en las bases de datos del gran hermano, no se no se. Aparte de que no se yo si la seccion espanhola ha hecho sus deberes para que puedas conseguir las licencias a traves de ellos o vas a tener que matarte a pelear con ellos para que lo hagan, claro que tambien te puedes registrar en Encompass (Decus americana). Los listados del 95% estan disponibles por unos 500$ siempre que puedas demostrar un uso legitimo (el precio es aproximado y cambia de hoy para manhana)

- Los overflow de buffer, stack, etc... causan la muerte del proceso pero raramente te permiten ejecutar nada debido a varias cosas, principalmente la estructura de la memoria es distinta, las posiciones en memoria de las diversas estructuras no es fijo, sobre todo en las maquinas alpha y los procesos son lo que se llaman pesados, tiene un monton mas de contexto para poder ejecutar una CLI, o interfase de lenguaje de comandos,(en teoria podria ponerse una diferente para cada usuario, en la practica es DCL, lenguaje de comandos Digital), y el proceso que crea todo ese contexto LOGINOUT realiza la autorizacion que te pediria tu username y password. (si quieres la informacion completa te buscas un libro, en la actualidad una serie, estructuras e interioridades de VMS de Digital Press naturalmente esa es la traduccion del titulo del libro del ingles), naturalmente hay quien dice que el sabe como hacerlo o como conseguir plenos privilegios desde una cuenta de usuario basica, pero no te lo diran ni muerto.

  Tienes 4 niveles de proteccion de memoria cada una asociada a cada nivel de ejecucion, para cualquier cosa interesante necesitas estar en un estado de privilegios elevado para lo cual necesitas que tu aplicacion este instalada con privilegios y en si misma realiza el error de seguridad mientras esta en ese estado elevado.Para lo cual tienes en cualquier caso hacer un monton de llamadas al sistema.Combinado con que todas las llamadas
  a los servicios del sistema se hacen basicamente mediante descriptores. Es decir punteros contados. o lo que es lo mismo, un conjunto de 2 punteros el primero apunta al tamanho y el segundo a donde comienza tu estructura y claro lo de pasarse del tamanho autorizado no funciona muy bien, (en realidad el tamnho minimo son tres punteros, marcando el tercero el tipo de datos que contiene el descriptor).

- En vez de solo dos privilegios root o usuario como en unix hay unos 20-30 diferentes privilegios, naturalmente hay un grupo de ellos que si los consigues es facil conseguir el resto pero para la mayoria de las operaciones no es necesario disponer de esa clase de privilegios y hay privilegios disenhados para hacer los procesos de mantenimiento mas comunes sin permitirte tocar otras cosas. Los ficheros no se le asignan privilegios sino

que se instalan con privilegios es decir para que el ejecutable tenga privilegios hay que llamarlo a traves del sistema y cuando termina el sistema limpia detras. Lo que es lo mismo lo que te permite conseguir privilegios son fallos de disenho o de configuracion no de programacion, "en general". Dos casos muy conocidos se hayan relacionado con el MAIL (el program de correo estandard) asociados con el echo de que desde el se puede abrir un subproceso con control del CLI o shell para los unixeros aunque presenta serias diferencias de concepto.

Fallo de disenho:
En un principio el mail requeria instalarse con privilegios y permitia abrir a CLI desde el con lo que al abrir el subproceso entrabas en la CLI con privilegios. Este error se corrigio hace mucho tiempo.

Fallo de configuracion:
Este es algo mas complicado y se haya asociado a cuentas cautivas, es decir cuentas que solo pueden acceder a un menu, el problema surge si una de las funciones es el uso del mail si se utiliza desde el mail el editor TPU que permite abrir un subproceso a la linea de comandos a su vez, se puede convertir en una autentica pesadilla para configurarlo correctamente y que nadie pueda llegar a la linea de comandos como un usuario normal (el truco es restringir el numero de subprocesos que puede abrir el usuario y hacerle abrir el editor sin generar subproceso).

Este es probablemte el tipo de cuenta presente en la biblioteca mencionada en set-24

- Mas lindezas del sistema, viene de fabrica con un proceso de auditacion, uno de accounting y uno de intrusiones de este ultimo ya hablamos algo al principio.

  Se pueden poner alarmas asociadas con casi cualquier cosa y los ficheros de log se mantienen constantemente abiertos por una aplicacion residente a la que puedes accerder interactivamente solo atraves de ciertas llamadas
  el sistema incluye utilidades para interaccionar con ellas, las malas noticias no se pueden borrar o modificar entradas. Lo que puedes hacer una vez tienes los privilegios suficientes es decirle que cree nuevos ficheros y borrar los antiguos.

  Ahora las buenas noticias, Estas maquinas suelen mantenerse practicamente solas asi que si encuentras una, especialmente en la universidad, es mas que probable que el administrador no este prestando mucha atencion, aunque si lo intentas con la base de datos de un banco no creo que tengas tanta suerte.

- Una vez entras en el sistema recuerda tienes los dias contados antes de que sea necesario cambiar el password (90 dias entre cambios es el estandard) no puedes repetir el antiguo password (mantiene una historia de los mismos) y mantiene un diccionario de paswords que no puedes usar (el por defecto es de palabras en ingles, lo cual hace la vida mas facil a los espanholes a la hora de elegir pasword).

- Algo mas a apreciar, existen basicamente 4 TCPIPs Multinet, TCPware, UCX y Compaq TCPIP. Estos dos ultimos son los Compaq-Digital pero en la version 5.0 de UCX cambio de nombre y basicamente de todo, las utilidades de configuracion siguen siendo las mismas pero la stack de comunicaciones se basa en la de Tru64 el Unix de Compaq para maquinas Alpha pero usando el sistema de memoria de VMS. Y las dos primeras pertenecen en este momento a la misma companhia con lo que el stack de comunicaciones se va pareciendo mas y mas cada dia.Y todas ellas tienen sus peculiaridades o lo puedes hacer aun mas divertido puedes estar usando Decnet sobre IP mediante Decnet-Plus El protocolo de Decnet funciona directamente sobre IP o puedes tunelearlo sobre IP mediante Multinet en cuyo caso puedes estar operando con el antiguo

aunque aun altamente apreciado Decnet-IV. Pensabas que lo de protocolos era complicado en UNIX. No nos olvidemos de LAT un precioso protocolo para terminales. Una X-Terminal sobre LAT es en torno a un 15-20% mas rapida que una sobre TCPIP el protocolo fue disenhado para minimo overhead, minima latencia y minimo Stack el resultado es que no es routable (menor overhead), No soporta lineas con mucha latencia, lineas virtuales y cosas asi, y que deja mas recursos en la X-terminal o NC para encargarse de procesar la imagen o lo que es lo mismo en una WAN no te servira de nada pero en una LAN es un encanto de tener.

- una observacion tambien puedes tener la CMU-ip una pila de TCP/IP gratuita y con la fuente disponible , aunque en la actualidad no muy usada y bastante anticuada, salvo que alguien este intentando conectar una antigua maquina para salvarla del desguace

EL TOMATE.

Lo que se ha visto hasta ahora, lo marca como un sistema no-Unix pero nada lo suficientemente diferente para merecer la pena el esfuerzo de leer esto salvo que te gusten el saber por el saber. Vamos con las diferencias Gordas.

RMS (Record Management System) A diferencia de unix el systema de acceso de ficheros es record orientado o lo que es lo mismo los ficheros no son una secuencia de bit (al menos no siempre) eso incluye la presencia en el sistema de fichero de ficheros indexados con las correspondiente extensiones de acceso de ficheros que proporcionan a todos los leguanjes de programacion como un ACCESS de microsoft, (todos los leguajes salvo C que presupone que el fichero es una secuencia de bits).

LOGICALS. o nombres logicos son como las variables de ambiente de unix pero mas amplias tienes 4 niveles trabajo, proceso, grupo y sistema y con varias propiedades extras. Aparte en cada nivel puedes crear tablas de variables expecificas para tus usos especificos asi es practica comun desarrollar el sistema de configuracion de los programas como listas de logicals al nivel adecuado.

SYMBOLOS seria el equivalente a los alias de UNIX.

CLUSTERS. Aqui es donde las cosas enpiezan a ponerse interesantes. El punto interesante es la presencia del DLM o Distributed Lock Manager o un sistema de lock distribuido por todo el Cluster que le permite por ejemplo acceder a varias maquinas los mismos discos al mismo tiempo. Me explico el proceso general en un sistema unix o NT es una maquina procesa las solicitudes para el disco y la otra maquina aunque conectada fisicamente al disco no realiza operaciones sobre el mismo directamente sino solo a traves del primer ordenador si quieres tener ese segundo como backup necesitas que continuamente este enviando la informacion de lo que esta haciendo para que el segundo pueda tomar sobre las operaciones de primero si este falla.(En las ultimas versiones Cluster de Solaris, por ejemplo, varias maquinas pueden acceder directamente a un disco pero solo una maquina se encarga de controlar(leer escribir) los metadata. En VMS ambos sistemas pueden acceder fisicamente al disco al mismo tiempo y se coordinan a traves del DLM. La informacion que antes se pasaba solo para mantener el log ahora es usada para que los diversos nodos puedan acceder al disco y la informacion que antes debia ser pasada al nodo principal del segundo nodo para ponerla en el disco ahora es pasada directamente al disco.

Naturalmente no hay una comida gratis, estas pasando menos informacion de un nodo a otro pero ahora hay que manejar esos locks consumiendo CPU, memoria las ventajas para un nodo aislado, no son importante, para solo dos nodos y los discos sigue siendo pequenhas, el punto es que Compaq soporta hasta 96 nodos en un cluster (aqui soportar significa que no tienes que

pagarle mas a Compaq por tener los 96 nodos separados que en un cluster para que venga corriendo a arreglar algo si hiciste una pifia). Las utilidades de control del cluster es decir las utilidades que te permiten controlar todo el cluster o trozos del mismo como si de una unica maquina se tratara estan limitados a 128 nodos es decir si tienes mas de 128 nodos tendras que empezar a manejar los nodos en grupos y hace la vida mas dificil y menos automatizable. Y luego tienes los limites de las estructuras en los sistemas (Internet Rumor: las malas lenguas dicen que han existido un par de clusters de aprox. 150 y algunos dicen que los han manejado pero nadie dice donde ni cuando) se hayan limitadas a unos 256 nodos pero seguro que al igual que las utilidades de manejo, en realidad es una y es llamada sysman, que la casca a 128 si seguimos subiendo algo mas la cascara.

Para hacer todo esto posible es necesario hablar entre las maquinas lo suficientemente rapido para que saber si alguien esta usando algo no tome tanto tiempo que haga todo el tema inutilizable para eso tenemos como no otro protocolo especializado el SCS otro protocolo disenhado para minima latencia, maxima reavilidad (perdon por la inglesada quiero decir que la maxima seguridad de que el paquete que envias llega y que el menor numero de paquetes hay que volverlo a enviar porque se perdieron) que de nuevo es no enrutable pero a cambio te permite tener con una linea de comunicacion dedicada nodos hasta unos 200 Km puede funcionar sobre CI (cluster interconexion), FDDI, Ethernet, Frame relay, ATM, memory Channel, memoria compartida, y ahora se trabaja en Fibre Channel.

Esta ultima cualidad es la que realmente le gana mucha popularidad entre sitios como bancos y bolsas puedes poner un segundo centro de proceso a 200 Km pones los discos mirror en cada uno de ellos, contratas un par de lineas dedicadas entre los dos sitios y a diferencia del tipico hot backup el traspaso puede ser tan corto como un par de segundos, en caso de algun problema, en alguno de los sitios y puedes usar las maquinas en los dos sitios al mismo tiempo, mientras los dos sitios esten en perfectas condiciones.

Algo realmente lindo seria un centro en Madrid otro en Zaragoza y otro en Burgos (SE dice que con lineas especiales la cosa se amplia hasta los 500 Km). Menudo desastre natural necesitas para que te tiren el sitio abajo. Al operar como una sola maquina permite que si le pones el DNS de si mismo bajo su propio control sabe cual es la maquina menos ocupada y enviara las solicitudas a esa. Dado que todos los nodos pueden acceder a todos los recursos al mismo tiempo se puede configurar para no permitir acceso a una maquina actualizar el sistema operativo desde ella reboot para la nueva version. Y segun vayas teniendo oportunidad de reboot las demas maquinas iran llendo a la nueva version sin que nadie se entere de lo que estas haciendo.

Que para que quieres saber todo esto, el hecho de que el cluster sea accesible como cluster no significa que no lo este tambien como cada nodo independiente. Los usuarios estan generalmente definidos a nivel de cluster y no a nivel de cada maquina pero no obligatoriamente. Piensalo un poco y te daras cuenta de los inusuales configuraciones que puedes encontrar, sobre todo si el administrador es un pelin nulo.

Una buena primera aproximacion a diferentes tipos de cluster puede encontrarse en.

http://www.infoworld.com/articles/fe/xml/01/12/17/011217feclustertca.xml

ACLs

O Access Control List, listas de control de accesso. Existen 4 privilegios basicos. Read, Write, Execute, Delete. Ejecutar incluye leer y borrar incluye escribir. (el sistema de ficheros es versionado asi que cuando escribes creas un nuevo fichero con una nueva version con lo que escribir algo en un fichero

no borra la informacion en el fichero anterior, si tienes impuesto un limite de versiones para el fichero el Write privilegios solo te permite escribir nuevos ficheros hasta que has completado el numero de versiones permitido despues necesitaras a alguien con Delete para eliminar las versiones antiguas). No todo es un fichero asi que tienes otros privilegios tales como Control, Management, Acesss y a todo se le puede imponer dichas listas de acceso, aparte de incluir ACL binarios es decir controlados por el sistema pero que una aplicacion expecifica puede usar para lo que quiera.

LOS DEFECTOS

Se parece muchisimo a Unix pero no lo suficiente. El proceso basico es tan ligero como en unix pero si le incluyes el contexto, logicals, shell(CLI)... se vuelven muy pesado lo que combinado con otras pequenhas diferencias lo hacen mas adecuado para procesos en hilos, aunque su estructura basica se realiza mediante procesos ejecutandose dentro de un mismo contexto comunicandose asincronamente y no mediante forks (forks /exe funcionan no asi los fork a secas) El x-server conocido como DECwindows (motif o CDE) se ejecuta como un proceso de usuario que este en ese momento conectado y por tanto se haya limitado a las quotas de memoria del usuario (oh por defecto los usuarios tienen quotas para cualquier cosa que puedas imaginar) lo cual puede hacer muy lento para ciertos usuarios el xserver asi como dadas las diferencias en el systema algunas partes de las API de los xserver que el estandar deja al gusto del disenhador presentas diferencias mayores que las comunes entre unixes. Conseguir I/O asincrono es mucho mas facil que en unix (todo el systema esta basado en comunicaciones asyncronas) salvo a la hora de escribir en disco. En ese caso la seguridad prima sobre la velocidad por defecto,asi que por defecto no se devuelve acceso al proceso hasta que la informacion se ha escrito fisicamente a disco.

Combinado con que los valores por defecto de los caches de lectura fueron disenhados cuando la memoria era muy cara lo que los hace los valores por defecto muy pequenhos para lo habitual en la actualidad.

Eso tiene dos efectos, salvo procesos con un gran numero de forks es facil pasar pogramas de unix a VMS pero en el proceso si no se optimizan para sus peculiaridades se vuelve mucho mas lento en general, en especial si el proceso unix utiliza gran cantidad de pequenhos ficheros que se escriben y se leen repetidas veces, para pasar informacion, en unix el procedimiento estandar es que el fichero puede nunca llegar a tocar el disco quedandose en memoria cache, no asi en VMS.

En la actualidad existen varios proyectos para incrementar la facilidad con que se puede portar software de unix a VMS manteniendo la misma eficiencia.
Si tendran exito, quien sabe? ya se vera.

Un caso seria una applicacion residente que tiene que releer los ficheros de configuracion de vez en cuando para asegurarse que no se han producido cambios. en unix se crea un fichero que se relee regularmente, el fichero se accede regularmente asi que salvo que estes muy corto de memoria permanecera en el cache hasta que alguien lo cambie con un editor (aun asi el primer cambio se producira en el cache), ese proceso seria muy costoso en VMS la forma correcta seria una lista de logicals definidos en la tabla de adecuada.
Esto en si mismo no es ni mejor ni peor simplemente diferente pero te permite algunas cosas que pueden ser bastante dificiles con ficheros.

Esto te puede permitir por ejemplo que la accion se realice de forma diferente segun quien se conecte o segun a que grupo pertenecca (se puede hacer la busqueda del correspondiente logical en una lista de tablas de forma muy sencilla y sin incrementar los recursos necesarios para realizarlo de forma apreciable).Para crear un lock en algun recurso es habitual en unix

crear un fichero vacio para que otros programas que van a trabajar
en conjuncion puedan comprobar si tu lo estas utilizando, en VMS la
respuesta es solicitar un lock al DLM que para eso esta pero al pasar
un programa desde unix eso supone un cambio no standard que es imposible
pasar de vuelta al codigo fuente unix.

La actual cancelacion del futuro desarrollo de nuevos procesadores ALpha tras
el EV7. Ha dejado el futuro de este SO dependiente de su traslado a los
sistemas IA-64 de intel y el futuro de dichos sistemas, aunque se ha dicho
que se esta aprovechando este segundo traslado a un nuevo tipo de maquinas
para eliminar las dependencias de harware y poner todas las funciones en
memoria, en principio solo necesitaria procesadores con 4 anillos de ejecucion.

****

Esto seria una primera aproximacion a lo que es VMS.

Si le quereis echar un vistazo al modelo de seguridad de VMS podeis visitar

http://www.blacksheepnetworks.com/security/resources/openvms/

O leeros la guia de seguridad de la documentacion de VMS
(si ya dije que lo que no documenten).

En Defcon9 se presento un nodo VMS como bastion, con cuentas gratuita, servidor
Web, telnet y FTP y permanecio inviolado, desgraciadamente las reglas
cambiaron en el defcon10 eliminando que se pudiera volver a presentar.
****

Seguro que tengo un par de errores y equivocaciones en lo escrito, pero errar
es humano y lo de leerte 20000 paginas de documentacion es excesivo.
****

<SET comments>
Para probar si alguien le pica la curiosidad,.....

telnet.ucc.edu (151.198.62.71)

http://www2.wku.edu/www/vmsguide/chap1.html

http://www.testdrive.compaq.com/

http://home.wanadoo.nl/erens/openvms/

http://www.hobbesnet.org/
Se puede uno registrar. Te envian la passsword por correo.
No tienen grandes medios,
 MicroVAX 3100 model 40
 VAXstation 4000 model 60
 VAXstation 4000 VLC

telnet://ludens.elte.hu/ y login como GUEST

EOF

-[0x09]--
-[LSSI]--
-[by anonimo]---SET-29--

La LSSI: un resumen

0- Prefacio
==========
Un comentario de web@set-ezine.org, y el que una de las peticiones mas insistentes de SET sea 'un articulo legal con fundamento', me ha llevado a redactar el presente texto.

No soy experto en Derecho y ya os digo yo que este texto NO tiene fundamento. Lo que si tiene es un resumen de la LSSI: la Ley 34/2002, de 11 de julio, de servicios de la sociedad de la informacion y de comercio electronico.

MUY importante (!!!): Este texto contiene muchas imprecisiones, porque las leyes jamas se resumen (cada letra y cada coma de la ley tiene su significado).

Esta pensado para aquellos que no tengan ganas de leer <mal hecho!!> sus 45 articulos y sus 17 disposiciones (160 kb en formato HTML, unas 3-6 horas de lectura).
El resumen se desglosa de forma similar a la ley, donde el numero de apartado de este texto equivale al Titulo de la LSSI.
Tambien pretende estimular al lector a leer la LSSI entera y cualquier otra norma legal, por 3 motivos:
1- Conocer tus derechos y obligaciones.
2- Hay textos legales que contienen informacion tecnica muy interesante.
3- No se deberia estar a favor ni en contra de una ley sin leersela integramente, para librarse de los propios prejuicios y de las influencias de terceros (excepto si son de tu abogado ;).

Al final, unas reflexiones.

A los lectores de fuera de Espana les encomiendo a que esbocen la legislacion de su pais.

Una ultima puntualizacion muy importante (!!!):
La LSSI no es la unica norma legal que regula los 'servicios de la sociedad de la informacion', ya que hay mucha (muchaaaa!!!) legislacion que tiene disposiciones sobre esta materia; legislacion sobre: datos personales, firma electronica, telecomunicaciones, seguridad, salud publica, defensa, consumo, competencia y propiedad intelectual, tributacion, contratos, comercio...
En caso de duda, consulte con su abogado.

Empezamos...

1. ¨A quien se aplica la LSSI?
==============================
Este apartado es el mas espeso, pero si se resume pierde su valor. Animo.

1.1 Servicios establecidos en Espana

La LSSI se aplica a las personas y servicios que esten o no esten en Espana, siempre que presten 'servicios establecidos en Espana'.

¨Que es 'servicios' y que es 'establecidos en Espana' para la LSSI? Preparate:

- 'Servicio de la sociedad de la informacion' (para abreviar, 'e-Servicio'):
  es la contratacion de bienes y servicios por via electronica (compra,
  alquiler, subhasta, apuestas...), el suministro de informacion por dicho medio
  (publicaciones, como las e-zines), las actividades de intermediacion relativas
  a la provision de acceso a la red, a la transmision de datos por redes de
  telecomunicaciones, a la realizacion de copia temporal de las paginas de
  Internet solicitadas por los usuarios, al alojamiento en los propios
  servidores de informacion, servicios o aplicaciones facilitados por otros o a
  la provision de instrumentos de busqueda o de enlaces a otros sitios de
  Internet, asi como cualquier otro servicio que se preste a peticion individual
  de los usuarios (descarga de archivos de video o audio...), SIEMPRE QUE
  REPRESENTE UNA ACTIVIDAD ECONOMICA PARA EL PRESTADOR, aunque el destinatario
  no pague por ello.

Estos servicios son ofrecidos por los operadores de telecomunicaciones, los proveedores de acceso a Internet, los portales, los motores de busqueda o cualquier otro sujeto que disponga de un sitio en Internet a traves del que realice alguna de las actividades indicadas, incluido el comercio electronico.

No son e-Servicio sujetos a la LSSI:
- servicios prestados por telefonia, fax o telex.
- intercambio de e-mails o equivalente para fines ajenos a la actividad
  economica.
- servicios de radiodifusion (tele o radio).
- teletexto televisivo.
- Servicios de notaria y registro de la propiedad y mercantil.
- Servicios de procuradores y abogados en caso de representacion y defensa en
  juicio.
- Servicios relacionados con medicamentos y productos sanitarios.

Buf! El otro concepto es mas cortito:

- 'Establecido en Espana': el lugar desde el que se dirige y gestiona la
  actividad economica esta situado en Espana.
  El que los medios tecnologicos utilizados para dicha actividad esten situados
  en Espana no siempre significara que el prestador del e-Servicio este
  'establecido en Espana'.

¨Que? ¨Te han incluido? Seguramente no. Y si estas fuera de Espana te interesaran los siguientes apartados 1.2 y 1.3.

1.2 Servicios establecidos en la UE

Tambien se aplica a los prestadores de e-Servicios establecidos en otro estado de la Union Europea (UE) cuando el destinatario este en Espana y ademas el servicio afecte a:
- Derechos de propiedad industrial o intelectual. O sea, espionaje industrial
  y derechos de autor.
- Emision de publicidad por instituciones de inversion colectiva. ¨Bancos?
  ¨Bolsa?
- Actividad de seguro directo realizada en regimen de derecho de
  establecimiento o en regimen de libre prestacion de servicios. ¨Seguros?
- Obligaciones nacidas de los contratos celebrados por los consumidores.
- Regimen de eleccion de la legislacion aplicable a un contrato.
- Licitud de las comunicaciones comerciales por correo electronico o
  equivalente no solicitadas. O sea, publicidad que nadie pidio.

1.3 Servicios establecidos fuera de la UE

Tambien se aplica a los prestadores de e-Servicios establecidos fuera de la UE [hola America] cuando el e-Servicio vaya dirigido especificamente a Espana o cuando el servicio atente contra la seguridad, salud, derechos humanos... (ver ap.2).
No incluyen aquellos que indiquen los tratados internacionales (...).

2. Y esos e-Servicios, ¨pueden ser de cualquier tipo?
==========
No, hay restricciones.

Afortunadamente existe el principio de libre prestacion de e-Servicios, incluso si provienen de la UE, y no necesitas autorizacion para prestarlos.

Aunque si proviene de fuera de la UE se regula por el tratado internacional que toque.

NO se puede prestar el e-Servicio si atenta o puede atentar contra los siguientes principios:

a) El orden publico, la investigacion penal, la seguridad publica y la defensa nacional.
b) Proteccion de la salud publica, consumidores y usuarios.
c) Respeto a la igualdad y dignidad de las personas.
d) Proteccion de la juventud y de la infancia.

Yo creo que aqui se incluyen webs que venden bombas, droga, pornografia infantil, informacion restringida, servicios robados o fraudulentos...
En mi opinion, las webs que ofrecen informacion sobre esos temas, si la ofrecen sin hacer negocio por ello, no estan sujetas a la LSSI. ¨No?

3. ¨Los prestadores de e-Servicios tienen alguna obligacion?
==========
Pues claro. Aqui esta el principal conflicto, ya que se confunde con los derechos de libre informacion, secreto de las comunicaciones y proteccion de la intimidad y de los datos personales.
Esto recorta las alas del anonimato, uno de los principios intrinsecos de Internet.

Ademas de lo que digan otras leyes, la LSSI especifica lo siguiente:

3.1. Registro del nombre de dominio

La direccion de Internet del prestador del e-Servicio establecido en Espana se anotara en el registro publico (generalmente el Registro Mercantil, y antes de 1 mes).

Nota: Al final de la LSSI se trata la extension ’.es’, la entidad ’Red.es’ y el ’Plan Nacional de Nombres de Dominio de Internet’. Consultad las referencias para mas informacion.

3.2. Informacion general

El prestador del e-Servicio debera disponer de medios (p.ej. su propia web) para que Administracion y ciudadanos puedan vincular la direccion de Internet con el establecimiento fisico y con el prestador del e-Servicio, requiriendose

la siguiente informacion:

1- Nombre o razon social, domicilio, e-mail o equivalente.
2- Datos del Registro.
3- Si la actividad requiere autorizacion administrativa, datos relativos a esta autorizacion.
4- Si ejerce una profesion regulada: Colegio y numero de colegiado, Titulo academico o profesional, Estado que expidio el titulo y homologacion si la hay, Normas aplicables al ejercicio de su profesion o los medios para conocerlas.
5- NIF.
6- Precio del servicio o producto, precisando gastos de envio e impuestos.
7- Codigos de conducta a los que se adhiere, o los medios para conocerlos (ap.3.6).

3.3. Deber de Colaboracion

Los prestadores de e-Servicios de intermediacion (o sea, proveedores de acceso a la red, de transmision de datos, de alojamiento en servidores, de buscadores, de enlaces...), ademas de registrarse y de ofrecer informacion sobre ellos, estan obligados a:

- suspender la transmision,
- suspender el alojamiento de datos,
- suspender el acceso a las redes de telecomunicaciones,
- prestar cualquier otro servicio equivalente de intermediacion,

, cuando se lo exija la autoridad (policia, jueces, inspectores...).

3.4. Deber de retencion de datos de trafico

Los operadores y servicios de red o de comunicaciones, los proveedores de acceso a red, y los prestadores de servicio de alojamiento de datos deberan retener los datos de conexion y trafico generados por el e-Servicio por un maximo de 12 meses.

Esos datos seran solo los necesarios para facilitar la localizacion del terminal del usuario.

Estos operadores, servicios y proveedores no podran utilizar los datos para fines ilicitos. Y deberan evitar intrusiones y alteraciones de esos datos.

Los datos podran ser utilizados por autoridades judiciales, policiales y militares en investigaciones criminales o para proteger la seguridad publica o la defensa nacional.
En mi opinion, este punto es delicadisimo...

3.5. ¨Y se es responsable por lo que otro haga?

En principio no.

Los prestadores de e-Servicios estan sujetos a responsabilidad civil, penal y administrativa (o sea, se les puede demandar, abrir una querella criminal o abrirles un expediente administrativo, respect., por violar la LSSI).

Los operadores de redes y proveedores de acceso NO son responsables de la informacion transmitida o almacenada por ellos (excepto si la manipulan o distribuyen de forma ilicita).

Los prestadores de servicios que realizan copia temporal de los datos solicitados por los usuarios, NO son responsables de la informacion transmitida o almacenada por ellos (excepto si la manipulan o distribuyen de forma ilicita). Han de retirarla cuando el lugar de la red en que se encontraba inicialmente la retire.

Los prestadores de servicios de almacenamiento de datos NO seran responsables de esos datos.
Los prestadores de servicios que faciliten enlaces a contenidos o instrumentos de busqueda NO son responsables por la informacion a la que se dirije al usuario.
En estos 2 casos (almacen y enlace), se exceptua cuando llegan a conocer que son actividades o datos (1) ilicitos o (2) indemnizables por terceros.
En ese caso (si lo saben), tambien ellos seran responsables si no retiran o bloquean los datos (excepto si se mantienen bajo control de la autoridad).
¨Y como se sabe si lo saben? Se considera que lo saben cuando (1) la autoridad declara su ilicitud, o (2) cuando los prestadores del e-Servicio conozcan la resolucion de indemnizacion.

3.6. Codigos de conducta

La LSSI deja en manos de asociaciones y organizaciones su autoregulacion (o sea, la elaboracion y aplicacion de unos principios de actuacion en su sector) para:
- Detectar y retirar contenidos ilicitos.
- Proteger al usuario de comunicaciones comerciales no solicitadas.
- Resolver conflictos que surjan por la prestacion de e-Servicios, de forma extrajudicial.

Antes de octubre de 2003 deberia de haberse aprobado un distintivo para identificar prestadores de e-servicios que respeten codigos de conducta.

3.7. Comunicaciones comerciales por via electronica (CCVE)

Y ahora, vamos a publicidad y volvemos enseguida...

Dice la LSSI que las CCVE se han de identificar:

- Deberan ser claramente identificables como tales y deberan indicar la persona o entidad en nombre de la cual se realizan.

- En el caso de tener lugar por e-mail o equivalente incluiran al comienzo del mensaje la palabra ’publicidad’ (Buscadlo en vuestros mails recibidos).

- En el caso de promociones, descuentos, concursos... debe dejarse claro que son tales.

Tambien dice la LSSI que podemos no recibir CCVE:

Queda prohibido el envio de CCVE no solicitados o no autorizados por el destinatario. Se exceptua cuando hay un contrato previo y se envia informacion sobre productos similares al contratado.
En todo caso el prestador de e-Servicios ofrecera al destinatario la posibilidad de oponerse al tratamiento de sus datos con fines promocionales, de forma facil y gratuita.
El destinatario de la CCVE puede cancelarla si asi lo expresa al remitente.

Si los prestadores de e-servicios almacenan datos del usuario, le deben informar de ello y le deben dar la posibilidad de rechazar ese almacen.

Me parece que muchas empresas infringen este apartado enterito. Por no hablar de los banners publicitarios (¨acaso no son CCVE?).

4. Contratacion por via electronica

Veamos ahora que dice de los contratos por via electronica (para abreviar, e-contratos), como los que nos obligan a Aceptar antes de abrir una cuenta de e-mail, o al comprar algo por Internet, o al entrar en algunas webs...

La LSSI equipara los e-contratos a los contratos de papel.
Ademas, el soporte electronico se equipara al 'por escrito', y a la prueba en un juicio (ya veo la gabardina del teniente Colombo llena de diskettes ;).

La LSSI no se aplica cuando:
- Sean contratos relativos a familia y sucesiones (ni bodas ni testamentos por Internet, de momento).
- Sean contratos que requieran documento publico, jurisdiccional, notarial, registral o de las autoridades.

La parte contratante de la primera parte y la parte contratante de la segunda parte, podran acordar una tercera parte para archivar los documentos (por NO menos de 5 anos). Algo asi como un testigo.

La LSSI exige al prestador del e-Servicio mas obligaciones:

Antes de la contratacion:

El prestador de e-Servicios debera informar claramente al destinatario sobre:
- Tramites del contrato.
- Si va a archivar el contrato electronicamente, y si va a ser accesible.
- Medios para identificar y corregir errores en la introduccion de datos.
- Idiomas a utilizar.

Excepto si:
- o ambas partes asi lo acuerdan y ninguna de las 2 sea un consumidor,
- o se celebre el contrato via intercambio de e-mails o equivalentes.

Ademas debera ofrecer al destinatario las Condiciones Generales (las que nadie se lee porque son muy largas).

Despues de la contratacion:

El prestador de e-servicios debe confirmar al que contrata que ha recibido la aceptacion del contrato, por e-mail (antes de 24 horas) o por el medio utilizado en la contratacion (lo antes posible).
Excepto si:
- o ambas partes asi lo acuerdan y ninguna de las 2 sea un consumidor,
- o se celebre el contrato via intercambio de e-mails o equivalentes.

Lugar de celebracion del contrato:

Constara que es:
- el de la residencia del consumidor,
- el de residencia del prestador del servicio (en el caso de contratos entre profesionales).

5. Solucion de conflictos
==========
Tus obligaciones vs. mis derechos... el eterno conflicto.

Se podra interponer accion de cesacion contra las conductas contrarias a la LSSI que lesionen intereses.

¨Interpolar lo queee...?
La accion de cesacion pretende conseguir una sentencia para que el demandado detenga la conducta contraria a la LSSI, y para prohibir que continue, e incluso para evitar que empiece.

¨Y yo puedo interponer eso?
Pozzi. Pueden interponer accion de cesacion las personas, grupos y entidades cuyo interes se vea lesionado, asi como el Ministerio Fiscal y las autoridades de defensa del usuario.

¨Y solo existe la via judicial? Con lo lenta que va la Justicia...
Ponno. Puedes contratar a la mafia rumana para ajustar cuentas. No obstante, la LSSI te propone los arbitrajes y los procedimientos extrajudiciales de los Codigos de Conducta.

6. Informacion y control
==========
La LSSI pretende promover y mejorar el acceso a la e-tecnologia con las siguientes medidas.

Los destinatarios y prestadores de e-servicios podran acudir (p.ej., electronicamente) a las Administraciones para informarse sobre:
- Derechos y obligaciones de la contratacion electronica.
- Procedimientos de resolucion de conflictos.
- Autoridades y asociaciones que faciliten informacion o asistencia.

Antes del 2006, la Administracion debe hacer accesibles a discapacitados y ancianos sus paginas de Internet. [¨Mas abstracto no se puede ser?].

Se insta al Ministerio de Tecnologia a presentar un ’Plan Cuatrienal para el Desarrollo de la Sociedad de la Informacion’, para fomentar la educacion y uso de las tecnologias de la informacion.

Tambien se contempla un circuito de control tecnico de resoluciones judiciales y decisiones extrajudiciales relevantes que afecten a la prestacion de e-Servicios. Basicamente:
CGPJ y organos arbitrales --> Ministerio de Justicia --> Comision Europea

El principal encargado de controlar e inspeccionar a los prestadores de e-Servicios es el Ministerio de Ciencia y Tecnologia, pero no es el unico en algunos sectores.

Los prestadores de e-Servicios tienen la obligacion de colaborar con ellos, y de promover el acceso de discapacitados y ancianos al contenido digital.

7. Infracciones y sanciones
==========
Y ahora, veamos por que nos pueden castigar, y que castigos hay. (En este apartado los hipervinculos irian de maravilla!)

7.1. Infracciones muy graves

- Incumplir las ordenes dictadas por las autoridades en referencia a las restricciones (ap.2: orden publico, defensa, salud, dignidad, infancia...).

- Incumplir las ordenes dictadas por las autoridades en referencia a la obligacion de suspender los e-Servicios de transmision (ap.3.3: transmision, almacen de datos, acceso a red...).

- No retener los datos de trafico y conexion (ap.3.4).

- Utilizar los datos de trafico y conexion retenidos para fines ilicitos (ap.3.4).

7.2. Infracciones graves

- No ofrecer informacion sobre nombre, direccion ni precios del e-Servicio (ap.3.2).

- Enviar mas de 3 comunicaciones comerciales en 12 meses, por e-mail o equivalente, a destinatarios que no lo autorizaron o que solicitaron su cancelacion (ap.3.7).

- No ofrecer las Condiciones Generales de un e-contrato (ap.4).

- No confirmar de forma habitual la recepcion de aceptacion de un e-contrato cuando se estaba obligado a hacerlo (ap.4).

- No permitir la inspeccion a las autoridades (ap.6).

7.3. Infracciones leves

- No comunicar al Registro la direccion de Internet del e-Servicio (ap.3.1).

- No ofrecer informacion sobre datos del Registo, datos de autorizacion si los tiene, datos profesionales si esta obligado, NIF y Codigos de Conducta adscritos, del e-Servicio (ap.3.2).

- No identificar la publicidad (comunicaciones comerciales, ofertas y concursos) como indica la LSSI (ap.3.7).

- Enviar hasta 3 comunicaciones comerciales en 12 meses, por e-mail o equivalente, a destinatarios que no lo autorizaron o que solicitaron su cancelacion (ap.3.7).

- No informar antes de un e-contrato aquello a que la LSSI obliga (ap.4).

- No confirmar de forma puntual la recepcion de aceptacion de un e-contrato cuando se estaba obligado a hacerlo (ap.4).

7.4. Sanciones

- Por infracciones muy graves: multa de 150.001 hasta 600.000 euros. Si en 3 a-os se hacen firmes 2 o mas infracciones muy graves, se puede prohibir la actuacion en Espana por hasta 2 a-os.

- Por infracciones graves: multa de 30.001 a 150.000 euros.

- Por infracciones leves: multa de hasta 30.000 euros.

- Por infracciones graves y muy graves, segun la repercusion social, se podra hacer publica la sancion (en un Diario Oficial, en 2 periodicos, y en la web de inicio del sancionado).

- Si el prestador del e-Servicio esta establecido fuera de la UE, se podra impedir el acceso a ese e-Servicio desde Espana por un maximo de 24, 12 o 6 meses, segun sean infracciones muy graves, graves o leves, resp.

- Las sanciones se estableceran teniendo en cuenta la intencionalidad, duracion, reincidencia, da-os producidos, beneficios obtenidos por la infraccion, y volumen de facturacion afectada.

7.5. Medidas provisionales

¨Sanciones antes de la resolucion? Si, en casos extremos.

En el momento de abrir el expediente sancionador, y solo en infracciones graves y muy graves, se podra acordar:

- Suspension del e-Servicio y cierre de establecimiento,
- Precinto, deposito, incautacion de material,
- Advertencia al publico de la indisponibilidad por haber tramites sancionadores,

, todo ello con caracter temporal, hasta que no sea firme la resolucion.

Estas medidas provisionales tambien pueden tener lugar de forma urgente, antes de abrir el expediente. En ese caso, a los 15 dias se debe confirmar o cancelar.

Se impondra multa de hasta 6000 euros por cada dia que se incumplan las medidas provisionales acordadas.

7.6. Concurrencia sancionadora

Las sanciones de la LSSI son compatibles con otros procesos sancionadores por los mismos hechos, excepto con la sancion penal y con las sanciones establecidas en leyes sectoriales.
Y si hubiera un proceso penal ya abierto, el expediente administrativo (LSSI) no podra finalizar hasta recaer la sentencia judicial, y tampoco contradecirla.

7.7. Prescripcion

Existe un margen de tiempo pasado el cual la infraccion deja de serlo, y la sancion deja de tener valor, por ejemplo, por la lentitud de la Justicia:
- Infr. muy grave: 36 meses.
- Infr. grave....: 24 meses.
- Infr. leve.....: 06 meses.
- Sancion por infr. muy grave: 36 meses.
- Sancion por infr. grave....: 24 meses.
- Sancion por infr. leve.....: 12 meses.

8. Reflexion

```
============
...Y acabo el resumen.

...Y empezo el examen.

Al que pille copiando lo suspendo. Silencio!


La LSSI no me ha dado respuestas a todas mis dudas. Tal vez porque esperaba
una ley sobre datos personales, recorte de libertades, derechos de autor...
mientras que realmente tiene un enfoque mas comercial (obligaciones y
controles sobre los prestadores de e-Servicios, e-contratos, e-publicidad...).

Me apuesto un CD del Fary (no pirateado) a que no eres capaz de responder
correctamente ni a la mitad de las siguientes preguntas sin mirar lo que has
leido hasta ahora:

- La LSSI ¨es justa o injusta? Argumentalo.
  Una pista: dicen que la Justicia es un poliedro, y que una de sus miles de
  caras es lo que tu consideras Justo, y que las otras caras son lo que los
  demas consideran Justo, y que todas sus caras son iguales, mi peque-o
  saltamontes...

- La LSSI, ¨Infringe los derechos de libre informacion, de proteccion de datos
  personales, del secreto de comunicaciones, y de inviolabilidad de la
  intimidad? La LSSI ¨Pone barreras a quienes quieran infringir precisamente
  esos derechos?

- ¨Se deberian poner mas y mejores controles a las Autoridades a la hora de
  acceder a nuestros datos? ¨Deberian ponerte mas y mejores controles a la
  hora de fisgar los datos de los demas? ¨Sabes diferenciar los conceptos de
  Justicia y de Impunidad?

- ¨Se tendria que castigar a las Autoridades por interceptar datos de un grupo
  terrorista cruel y sangriento (tal vez eviten una masacre de inocentes)?
  ¨Vale mas prevenir que curar, y donde esta el limite de la prevencion?
  ¨Que opinas de las guerras preventivas...? Espera!!! Esta la contesto yo:
  ---Follad como conejos!! ---Basta ya de guerras!!!.

- ¨Castigarias a un inspector de e-Servicios por sacar provecho ilicito de tus
  datos personales de suscripcion al Porno-Pics? ¨Le pondrias una denuncia?
  Bueno, vale, te da corte... ¨y si en vez del Porno-Pics fuera una Gaceta de
  Negocios, le denunciarias? ¨Las leyes deberian recortar mas la impunidad de
  ese inspector que infringe la LSSI? ¨O habria que darle mas libertad?

- ¨Has violado alguna vez los preceptos de la LSSI? En caso afirmativo, ¨es un
  acto justo o injusto? ¨Te han violado en terminos de la LSSI? ¨Te gustaria
  que violaran tus derechos que la LSSI pretende defenderte? (Abstenerse de
  contestar los sado-masoquistas...).

- Si crees que no nos deben recortar tanto nuestros derechos (y por tanto no
  reniegas de tus derechos), ¨que alternativa a la LSSI propones para proteger
  esos derechos de los que no reniegas? Pista: criticar es muy facil (y
  destruir mas facil).

- ¨Crees que te mereces el CD del Fary? No!!! Te suspendo el examen!!! Tus
  respuestas no las considero correctas.
  Pista: Quien hace la ley hace la trampa. Id con cuidado.



9. Referencias
```

==========
1.- http://noticias.juridicas.com
    Miles y miles de leyes gratis. Excelente.
2.- http://www.terra.es/personal/parabolo/villanos/lssi.htm
    Campa-a contra la LSSI con enlaces a otros paises. Con mucho fundamento.

EOF

```
-[ 0x0A ]-----------------------------------------------------------------
-[ Microcontroladores ]---------------------------------------------------
-[ by blackngel ]-----------------------------------------------SET-29--
```

blackngel_hack@hotmail.com

MICROCONTROLADORES

"Bienvenidos al mundo de lo real..."

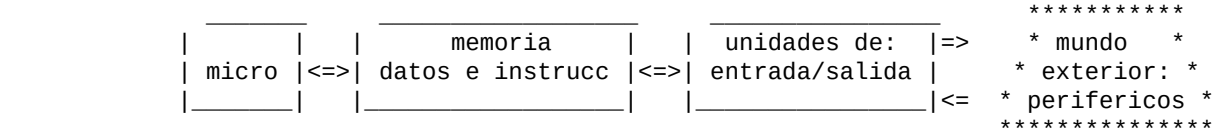
0.- INDICE

1.- INTRODUCCION

El control por ordenador presenta un problema: el sistema controlado tiene que estar siempre conectado al ordenador. Asi, un microondas o una lavadora tendrian que tener su propio ordenador, y un robot no podria alejarse mas de unos centimetros de la tarjeta controladora.

Afortunadamente, existen circuitos integrados que resuelven este problema. Permiten realizar gran numero de tareas y son muy parecidos a los ordenadores.

Un ordenador contiene un microprocesador que controla el resto de los elementos, una memoria en la que se almacenan datos e instrucciones, asi como una serie de tarjetas o unidades de entrada/salida encargadas de recibir
y enviar informacion a los perifericos.

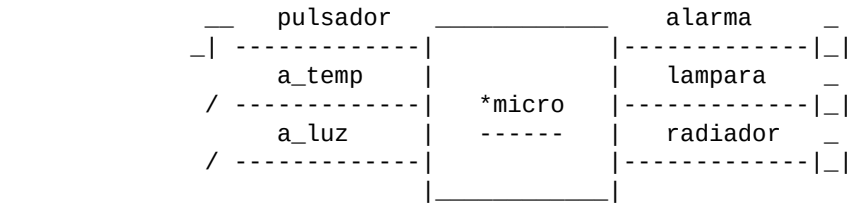


-Crees que es posible introducir un ordenador en un pequeÏo chip del tamaÏo de una uÏa? El desarrollo de la electronica ha permitido este prodigio.

Se denominan microcontroladores los circuitos integrados que contienen todos los elementos de un ordenador.

==========
¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢ 2.- APLICACIONES ¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
==========

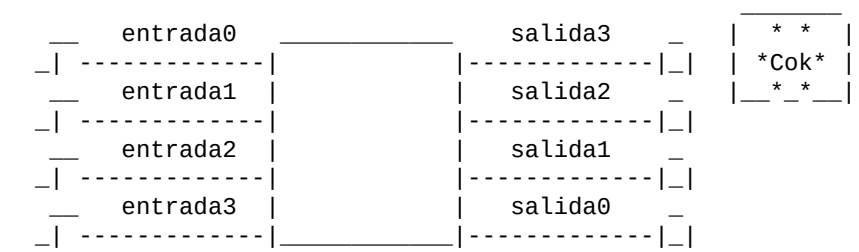
Los microcontroladores, tambien conocidos como PIC (controladores programables inteligentes), pueden obtener datos del exterior, ya sean analogicos (como la temperatura o intesidad de luz) o digitales (como el accionamiento de un pulsador), para despues procesar esos datos y accionar distintos dispositivos:



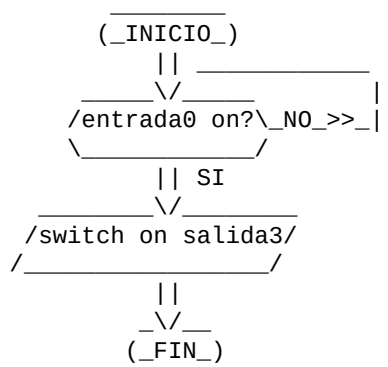
* Esquema del funcionamiento de un microcontrolador.

==========
¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢ 3.- CIRCUITO SENCILLO ¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
==========

Para entender como funcionan los microcontroladores,vamos a diseÏar una aplicacion muy sencilla: una maquina expendedora de bebidas.



Los microcontroladores, al igual que los ordenadores, pueden ser programados. El diagrama de flujo representa el programa que ejecuta el microcontrolador:



Fijate en el diagrama de flujo de la maquina expendedora de bebidas. En el, el programa esta continuamente pendiente de la entrada 0. Cuando se introduce una moneda (entrada0 on), se activa la salida 3 (switch on salida3), que entrega una lata de refresco. Una vez realizado esto, concluye el programa.

@@ PROGRAMANDO
@@@@@@@@@@@@@@

Los microcontroladores se programan en un lenguaje complejo denominado ensamblador. Tambien pueden programarse en Basic.

Para ello, una vez escrito el programa en un ordenador, se conecta el microcontrolador al ordenador a traves del puerto serie o paralelo, y se copia el programa en la memoria del microcontrolador.

Despues ya se puede desconectar y colocar en cualquier sistema que se desee controlar.

El programa que controla la maquina de bebidas seria el siguiente:

```
    IF Entrada0 = 1 THEN Lata
    GOTO Comienzo ’Ir a comienzo
    HICH Salida3 ’Salida3 = 1 --> Lata fuera
    STOP ’Fin del programa
```

Si lo pensamos bien, nos daremos cuenta de que despues de que la maquina de bebidas expulse la lata, la Salida3 continuara abierta, pues bien, la instruccion en este caso para desactivarla antes de volver al comienzo seria ’LOW Salida3’.

==========
£££££££££££££££££££ 4.- CROCODILE TECHNOLOGY £££££££££££££££££££££££
==========

Este programa permite simular circuitos con microcontroladores. De esta manera, es posible verificar su funcionamiento antes de montarlos.
Puedes conseguir una version de evaluacion del programa Crocodile Technology en la siguiente direccion: "www.crocodile-clips.com".

Los pasos que hay que seguir para realizar un montaje son los descritos a continuacion:

** En la barra de herramientas seleccionamos Componentes digitales.

** Aparece una nueva barra:
     - Los cinco botones centrales representan respectivamente: Elementos de entrada, Puertas logicas, Microcontroladores, Circuitos integrados y lementos de salida.

** Comenzamos colocando el microcontrolador. Con este fin, seleccionamos la opcion Microcontroladores > AÏadir microcontrolador. Una vez colocado en la
barra de herramientas, seleccionamos 18 pin PIC y en Opciones avanzadas escribimos el nombre y numero de patillas de entrada y salida:

    Label -> in0:   Entrada0

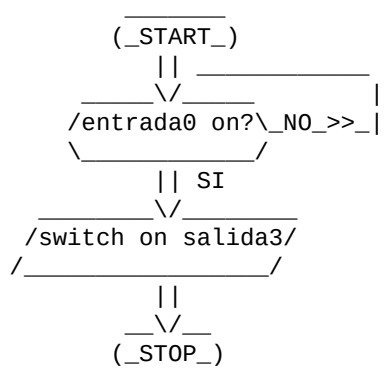
```
    Visible inputs:    Digital -> 4
    Visible outputs            -> 4
```

** Para aÏadir el detector de monedas, tiene que activarse el boton Elementos de entrada, con lo que aparece la siguiente barra de herramientas y en ella seleccionamos el detector de monedas y lo conectamos a la entrada0.

** Para terminar el circuito, solamente queda aÏadir el expendedor de latas. Con este fin, hacemos clic sobre el boton de Elementos de salida, seleccionamos despues Expendedor de latas y lo conectamos a la salida3.

@@ PROGRAMACION DEL MICROCONTROLADOR
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Con objeto de realizar el diagrama de flujo que va a ejecutar el microcontrolador en la barra de herramientas principal, seleccionamos Ver barra de diagramas de flujo. Apareceran los distintos elementos que pueden emplearse en el programa que estamos diseÏando.



Puedes probar el funcionamiento del microcontrolador haciendo clic sobre el boton |I>| (play). El programa se mantiene observando la entrada0. Si ahora introduces una moneda (haciendo clic sobre ella), aparecera un refresco.

-- Si quieres mantenerte entretenido y no aburrirte realiza un circuito controlado por microcontrolador que active una alarma al abrir una ventana.

==========
£££££££££££££££££££££ 5.- PROGRAMACION DE UN MICROCONTROLADOR £££££££££££
==========

Y por fin llegamos a la parte mas practica de todas, seguro que no os aburrireis, en esta seccion aprenderas a construir un circuito que, conectado al ordenador, te permita grabar los programas en un microcontrolador.

Tambien te familiarizaras con el empleo de un programa que lleve a cabo el transvase de informacion desde el ordenador hasta el microcontrolador.

@@ EL PROGRAMADOR
@@@@@@@@@@@@@@@@@@

Hay que escribir en el ordenador los programas que se van a almacenar en el microcontrolador. Para traspasarlos a este, es necesario un circuito que se conectara al ordenador a traves del puerto serie. Una vez grabado el

programa
en el microcontrolador ya podras conectar este en su circuito.

Para nuestros propositos utilizaremos el microcontrolador PIC 16F84 que podemos
encontrar en cualquier tienda de electronica.

El programador que presento a continuacion se puede comprar tambien en tu tienda de electronica habitual la unica diferencia es que normalmente en vez de venir equipado con un conector DB9 lo hara con uno DB25, para el que no le suene, este es el conector para el puerto paralelo, "si, el de la impresora". Este detalle no influira de ninguna manera en la programacion del PIC.

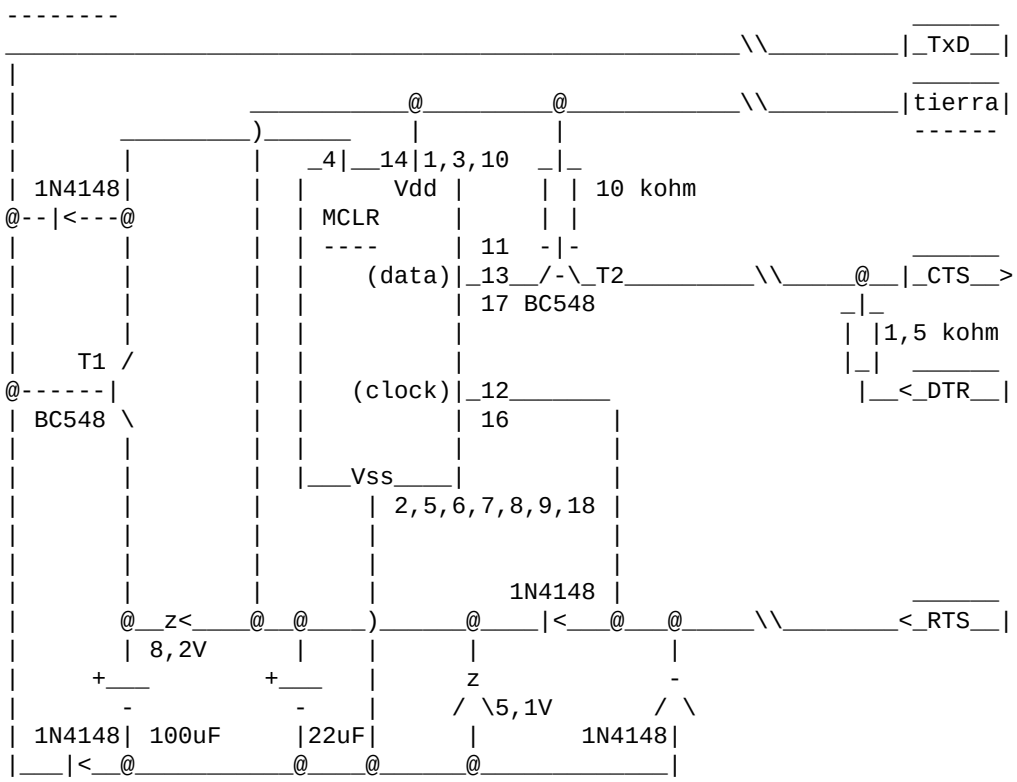
En los siguientes diagramas podras observar el circuito del programador, un esquema del conectar para el ordenador y el material necesario para realizar el montaje:

- Si quereis la imagen del circuito conectado al ordenador mailme a blackngel_hack@hotmail.com y os lo enviare sin ningun problema, tambien os puedo enviar ilustraciones del PIC 16F84 asi como de los diagramas que expuse anteriormente, pero seran de mejor compresion grafica.

MATERIALES
---

* 4 diodos 1N4148
* 2 diodos Zener (5,1 V y 8,2 V)
* 2 transistores BC548
* 2 resistencias (1,5 kohm y 10 kohm)
* 2 condensadores electroliticos (100 uF y 22 uF)
* 1 conector DB9 para el puerto serie.

CIRCUITO



- Mailme si quereis la imagen de este diagrama.

ACLARACIONES:

```
  |<  --> Diodos
  z<  --> Diodos zener
  @   --> Uniones de las pistas de cobre
```

CONECTOR DB9

```
                TxD DTR  _tierra
           _____|__|___/_
           \1o  o  o  o  o /
            \_o__o___o__o9/
                 |   |
               RTS CTS
```

CONECTOR DB25
---

* Por si las moscas, ademas lo importante es aprender, aprovechar la
   oportunidad que hoy estoy generoso en esquemas e informacion tecnica.

```
            / 13-------------------------1- \
           /    * * * * * * * * * * * * *   \
           \     * * * * * * * * * * * *    /
            \ --25--------------------14-- /
```

   (Este es el conector del puerto paralelo un conector DB-25)

Patilla	Nombre	Nivel	Entrada / Salida
1	Strobe	0	S
2	D0	1	S
3	D1	1	S
4	D2	1	S
5	D3	1	S
6	D4	1	S
7	D5	1	S
8	D6	1	S
9	D7	1	S
10	Acknowledge	0	E
11	Busy	1	E
12	Paper out	1	E
13	Select	1	E
14	Autofeed	0	S
15	Error	0	E
16	Initialize	0	S
17	Select Input	0	S
18	Gnd	/	/

%% INDICACIONES DE FUNCIONAMIENTO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Para programar el microcontrolador, necesitas una tension de entre 12 V y
14 V, que debes aplicar a la patilla 4 (MCLR). Esta tension se consigue
cargando los dos condensadores electroliticos.

Al utilizar los diodos Zener se obtiene una tension de aproximadamente 13 V

en la patilla 4 y de 5 V en la patilla 14.

La corriente de carga entra por TxD, llega a la base de T1, sale por el colector(con esta disposicion, la corriente de base sale por el emisor y el colector), pasa por los condensadores y retorna por RTS.

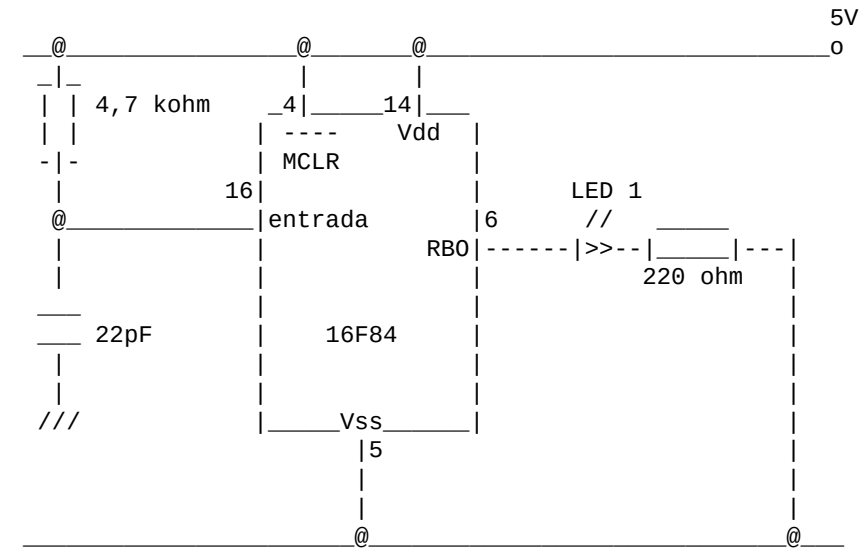
Los programas se envian al microcontrolador a traves de la linea DTR, y el microcontrolador los manda al ordenador a traves de CTS. El transistor T2 permite que se realicen ambas operaciones.

Es posible montar el programador anterior en una placa protoboard o en un soporte permanente (un circuito impreso), puesto que es un circuito que vas a utilizar con frecuencia para cualquier proyecto que quieras realizar con tu PIC.

@@ EL PRIMER CIRCUITO
@@@@@@@@@@@@@@@@@@@@@@

Con objeto de probar el programador recien creado y entender como se desarrolla una aplicacion mediante microcontrolador, vamos a desarrollar un circuito muy simple. Se trata de hacer que parpadee un LED.

Para ello, monta el siguiente circuito (a fin de que el circuito funcione, tienes que programar el microcontrolador 16F84 que hemos utilizado, asunto del que nos ocuparemos en el sigueinte paso).



@@ FASES DE PROGRAMACION
@@@@@@@@@@@@@@@@@@@@@@@@@

Una vez realizado el programador, hay que conectarlo al ordenador a traves del puerto serie y colocar el microcontrolador. A continuacion , para grabar el programa, debes seguir este procedimiento:

|1| EDITAR EL PROGRAMA

Si vas a realizar el programa en lenguaje ensamblador, usa un editor de textos simple (que no introduzca ningun codigo), como Bloc de notas.

Tras guardarlo como Prueba1.txt, sal del Bloc de notas, ve a la carpeta donde

se encuetnre el fichero y cambia el nombre a prueba1.asm.

La extension .asm significa <<assembler>> e indica que es un programa en lenguaje ensamblador.

El codigo es el siguiente:

```
   list p=16f84
   #include p16f84.inc
Tiempo EQU 22h
Vueltas EQU 23h
   org 0 ;origen del programa en memoria

CONFIGURAR
   BSF STATUS, RP0 ;iniciamos configuracion
   BCF TRISB, 0 ;RBO Salida
   BCF STATUS, RP0 ;fin de la configuracion

COMIENZO
   BSF PORTB, 0 ;encender LED
   CALL RETARDO ;ir a retardo
   BCF PORTB, 0 ;apagar LED
   CALL RETARDO
   GOTO COMIENZO ;ir a comienzo

RETARDO MOVLW b'11001000' ;Vueltas=200
        MOVWF Vueltas
MAS     DECFSZ Tiempo ;Tiempo=Tiempo-1
        GOTO MAS
        MOVLW b'11111111'
        MOVWF Tiempo
        DECFSZ Vueltas
        GOTO MAS
        RETURN
        END
```

- Lo que va despues de los ";" no son necesarios puesto que son comentarios.

Si vas a hacer el programa en Basic (lo que es recomendable, pues es mucho mas sencillo), puedes emplear tambien un editor de textos o el editor que acompaÏa al compilador Basic. El aspecto del programa es el siguiente:

```
bucle:   High 0       'Encender el Led conectadoa RB0
         Pause 500    'Esperar 500 ms

         Low 0        'Apagar el LED conectado a RB0
         Pause 500    'Esperar 500 ms

         GoTo bucle   'Ir a bucle
         End
```

- Lo que va despues de " ' " tambien funciona como comentario.

|2| COMPILAR EL PROGRAMA

En este punto tienes que transformar las instrucciones dadas en lenguaje ensamblador o Basic a ceros y unos, el unico lenguaje que son capaces de entender los circuitos digitales.

* Si se ha realizado el programa en lenguaje ensamblador, es posible utilizar el programa MPASM, que puede obtenerse gratuitamente en www.microchip.com. Selecciona las siguientes opciones:

    Source File Name: Direccion del archivo Prueba1.asm
    Radix: Default
    Warning Level: Default
    Hex Output: Default
    Generated Files: Error File y List File
    Macro Expansion: Default
    Processor: 16F84
    Tab Size: 8

A continuacion haz clic sobre el boton Assemble para obtener un fichero llamado Prueba1.hex, que contiene 0 y 1 que grabaras en el microcontrolador.

Aqui puedes observar el aspecto que tendria el fichero Prueba1.hex mediante el Bloc de notas:

```
_____________________________________________
:10000000831606108312061408200610082003280l |
:10001000C830A300A20B0A28FF30A200A30B0A28B5 |
:020020000800D6                              |
:00000001FF                                  |
_____________________________________________|
```

Observa que se utiliza el sistema hexadecimal para simplificar la representacion: como sabes, cada digito hexadecimal equivale a 4 bits.

* Si el programa ha sido realizado en Basic, es necesario un compilador de Basic.

   Uno de los mas conocidos y documentados es el compilador PicBasic o su version mas avanzada que recibe el nombre de PicBasic Pro. Mediante este compilador se obtiene tambien un fichero similar a Prueba1.hex.

|3| GRABAR EN EL MICROCONTROLADOR

Con objeto de grabar en el microcontrolador el programa Prueba1.hex (no importa si se ha obtenido partiendo de lenguaje ensamblador o de PicBasic), se usa un programa llamado IC-Prog (que tambien es gratuito y se puede descargar desde internet, ’www.ic-prog.com’).

En la pantalla inicial hacemos clic sobre Archivo > Abrir archivo y seleccionamos Prueba1.hex.

Antes de hacer clic sobre el icono Programar todo, que iniciara la descarga al microcontrolador, hay que seleccionar:

Ajustes > Opciones > Idioma > EspaÏol

Ajustes > Dispositivo > Microchip PIC > PIC16F84

Ajustes > Tipo hardware: JDM Programmer y el puerto serie al que esta conectado el programador.

Oscilador > RC y Bits de configuracion (WDT, PWRT y CP) a cero, sin seleccionar.

Despues ya es posible hacer clic sobre Programar todo.

Una vez programado el microcontrolador, se saca del programador y se traslada

al circuito (en nuestro caso el "LED parpadeador"), conectandolo correctamente prestando especialmente atencion a la colocacion de las patillas.

Al conectar la alimentacion, comenzara a ejecutarse el programa: El LED parpadeara continuamente.

** Ademas de hacer una programacion completa, IC-Prog tambien permite leer un
** dispositivo programado previamente, borrarlo para realizar otros programas
** o corregir errores y verificar si la programacion se ha realizado bien.
** Para mi esto es realmente algo muy interesante ya que podemos conectar
** microcontroladores de muy diversos aparatos y estudiar los programas que en
** ellos se encuentran e incluso modificarlos.

** Os daria ejemplos, pero me da miedo lo que podria llegar a hacer un
** microondas };.D

6.- DATOS TECNICOS

@@ INSTRUCCIONES

Aqui expongo el grupo de instrucciones especifico de nuestro microcontrolador "PIC 16F84A":

PIC16F84A INSTRUCTION SET

Operandos	Mnemonicos	Descripcion
	BYTE ORIENTED FILE REGISTER OPERATIONS	
f, d	ADDWF	Add W and f
f, d	ANDWF	AND W with f
f	CLRF	Clear f
!	CLRW	Clear W
f, d	COMF	Complement f
f, d	DECF	Decrement f
f, d	DECFSZ	Decrement f
f, d	INCF	Increment f
f, d	INCFSZ	Increment f, Skip if 0
f, d	IORWF	Inclusive OR W with f
f, d	MOVF	Move f
f	MOVWF	Move W to f
!	NOP	No Operation
f, d	RLF	Rotate Left f through Carry
f, d	RRF	Rotate Right f through Carry
f, d	SUBWF	Subtract W from f
f, d	SWAPF	Swap nibbles in f
f, d	XORWF	Exclusive OR W with f
	BIT ORIENTED FILE REGISTER OPERATIONS	
f, b	BCF	Bit Clear f
f, b	BSF	Bit Set f
f, b	BTFSC	Bit Test f, Skip if Clear

```
BTFSS        f, b      Bit Test f, Skip if Set

LITERAL AND CONTROL OPERATIONS

ADDLW        k         Add literal and W
ANDLW        k         AND literal with W
CALL         k         Call subroutine
CLRWDT       !         Clear Watchdog Timer
GOTO         k         Go to address
IORLW        k         Inclusive OR literal with W
MOVLW        k         Move literal to W
RETFIE       !         Return from interrupt
RETLW        k         Return with literal in W
RETURN       !         Return from Subroutine
SLEEP        !         Go into standby mode
SUBLW        k         Subtract W from literal
XORLW        k         Exclusive OR literal with W
```

Palabras reservadas para programar en MPASM
(Macro Pic ASM)

```
_ _BADRAM       - Identify Unimplemented RAM
BANKISEL        - Generate Indirect Bank Selecting Code
BANKSEL         - Generate Bank Selecting Code
CBLOCK          - Define a Block of Constants
CODE            - Begin an Object File Code Section

_ _CONFIG       - Set Processor Configuration Bits
CONSTANT        - Declare Symbol Constant
DA              - Store Strings in Program Memory
DATA            - Create Numeric and Text Data
DB              - Declare Data of One Byte
DE              - Declare EEPROM Data Byte
#DEFINE         - Define a Text Substitution Label
DT              - Define Table
DW              - Declare Data of One Word
ELSE            - Begin Alternative Assembly Block to IF
END             - End Program Block
ENDC            - End an Automatic Constant Block
ENDIF           - End Conditional Assembly Block
ENDM            - End a Macro Definition
ENDW            - End a While Loop
EQU             - Define an Assembler Constant
ERROR           - Issue an Error Message
ERRORLEVEL      - Set Message Level
EXITM           - Exit from a Macro
EXPAND          - Expand Macro Listing
EXTERN          - Declare an Externally Defined Label
FILL            - Specify Memory Fill Value

GLOBAL          - Export a Label
IDATA           - Begin an Object File Initialized Data Section
_ _IDLOCS       - Set Processor ID Locations
IF              - Begin Conditionally Assembled Code Block
IFDEF           - Execute If Symbol has Been Defined
IFNDEF          - Execute If Symbol has not Been Defined
INCLUDE         - Include Additional Source File
LIST            - Listing Options
```

```
   LOCAL           - Declare Local Macro Variable
   MACRO           - Declare Macro Definition
   _ _MAXRAM       - Define Maximum RAM Location
   MESSG           - Create User Defined Message
   NOEXPAND        - Turn off Macro Expansion
   NOLIST          - Turn off Listing Output
   ORG             - Set Program Origin
   PAGE            - Insert Listing Page Eject
   PAGESEL         - Generate Page Selecting Code
   PROCESSOR       - Set Processor Type
   RADIX           - Specify Default Radix
   RES             - Reserve Memory
   SET             - Define an Assembler Variable
   SPACE           - Insert Blank Listing Lines

   SUBTITLE        - Specify Program Subtitle
   TITLE           - Specify Program Title
   UDATA           - Begin an Object File Uninitialized Data Section
   UDATA_ACS       - Begin an Object File Access Uninitialized Data Section
   UDATA_OVR       - Begin an Object File Overlayed Uninitialized Data Section
   UDATA_SHR       - Begin an Object File Shared Uninitialized Data Section

   #UNDEFINE       - Delete a Substitution Label
   VARIABLE        - Declare Symbol Variable
   WHILE           - Perform Loop While Condition is True
   ____________________________________________________________________________
```

```
@@ FUSES
@@@@@@@@

Los fuses son unas variables que configuran ciertos aspectos de nuestro
microcontrolador. En el PIC 16F84A existen 4, a continuacion se explica
cada una.

1- OSC -> Tido de oscilacion ("tipo de reloj")

        * XT: Suministrado mediante un cristal de cuarzo de una frecuencia
              especifica.

        * RC: Oscilador indicado por una resistencia junto a un condensador.

        * HS: Como el primero pero para programas de alta velocidad.

        * LP: (Low Power) Lo mismo, pero la frecuencia no pasa de 200Khz.

2- WDT -> Este fuse le permite al PIC controlar el desarrollo de un bucle
           infinito y asi poder salir del mismo. Al entrar en el bucle y no
           borrarse el Watchdog por un tiempo, se provoca el reseteo del PIC.

3- PWRT -> Se retrase la inicializacion del PIC mientras se estabiliza la
            tension de alimentacion.

4- CP -> (Code Protection) Nadie tiene permiso para leer el codigo del PIC,
          pero sigue siendo posible su sobreescritura y ejecucion.


Si bien recordais estas variables son las que se pueden modificar atraves
del programa IC-Prog, pero si se desconoce por el momento su verdadera
utilidad es mejor no cambiarlos.
```

==========
£££££££££££££££££££ 7.- HISTORIA £££££££££££££££££££££££££££££££
==========

@@ ALAN TURING Y EL MICROCONTROLADOR
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Te sorprenderia saber la cantidad de aparatos que te rodean que estan controlados por ordenador: microondas, televisores, mandos a distancia, videos, camaras digitales, camaras de video, impresoras, lavadoras, telefonos moviles, el airbag del coche, el sistema antibloqueo de frenos, las alarmas, etc. Todos esos dispositivos incorporan un circuito integrado llamado microcontrolador, en cuyo interior hay un microprocesador y unidades de entrada y salida.

Los microcontroladores suponen la conjuncion perfecta entre informatica y electronica. Un ordenador concentrado en un pequeÏo circuito integrado que puede comunicarse con el exterior, almacenar datos y ejecutar distintos programas.

Los microcontroladores suponen la realizacion practica de los estudios efectuados tiempo atras por Alan Turing, y que fueron claves tambien para el desarrollo de los ordenadores. Durante sus aÏos de colegio e instituto, este matematico no destaco especialmente. Alrededor de 1945, sin embargo, concibio la idea de un ordenador con memoria que llevara a cabo operaciones aritmeticas y distintas tareas mediante programas, sin necesidad de otros componentes electronicos, Desde 1966, lleva su nombre uno de los premios tecnologicos mas importante, concedido por la Association for Computing Machinery.

@@ DOMOTICA
@@@@@@@@@@@

La domotica es la rama de la tecnologia que aplica los conocimientos de automatizacion y control para mejorar la vida en los domicilios particulares y en los edificios en general. Su objetivo es la construccion de viviendas inteligentes. Abarca cuatro campos fundamentales: la comodidad, la seguridad, el ahorro energetico y las comunicaciones. En las casas inteligentes, las luces se encienden y se apagan al entrar y salir de las habitaciones sin necesidad de interruptores, la concina electrica se desconecta cuando la comida esta preparada,y el frigorifico se encarga de ordenar los pedidos de la compra en incluso de sugerir el menu en funcion de los alimentos que contenga; es posible activar la calefaccion o dar la orden por telefono de preparar un baÏo cierto tiempo antes de llegar a casa; si decidimos ver una pelicula, basta con accionar el mando unico que controla toda la casa para que automaticamente, se bajen las persianas, se apaguen las luces y se activen el DVD y el televisor.

En el campo de la seguridad, las casas inteligentes controlan las instalaciones electrica, de agua y de gas, avisando de cualquier anomalia. Tambien velan por la seguridad de las personas mayores, enfermos y niÏos, y permiten observarlos en cualquier momento a traves de camaras web. Si detectan la presencia de algun intruso, bloquean todas las salidas y avisan a la policia.

%% CURIOSIDAD...
%%%%%%%%%%%%%%%

Existe un sofisticado cubierto que nos indica la temperatura de los alimentos

con solo pinchar en ellos. Se trata del tenedor inteligente. Ademas, este curioso artilugio se puede dejar clavado en el entrecot, el filete, etc, para no tener que estar pendiente de la sarten. Mediante radiofrecuencia, el tenedor inteligente nos comunica si la carne esta cruda, en su punto o bien hecha, en cualquier lugar de la casa en el que nos encontremos.

==========

£££££££££££££££££££££ 8.- DESPEDIDA £££££££££££££££££££££££££££££££

==========

@@ FUENTES DE INFORMACION
@@@@@@@@@@@@@@@@@@@@@@@@

- Tecnologia Secundaria Oxford Educacion
- Introduccion a los PICs by Ykappz
- www.crocodile-clips.com
- www.microchip.com
- www.ic-prog.com
- 7a69-14 Art. 9

Si quereis tener informacion para programar este mismo controlador en Linux os remito al articulo publicado en la e-zine 7a69 numero 14, articulo 9. Es peque¤o y tecnico pero tiene aspectos de lectura recomendable.

Bueno chic@s otro articulo mas para la coleccion, este tema es muy pero que muy interesante yo seguire estudiando por mi cuenta y realizando muchisimas mas pruebas, vosotros?, el poder esta en vuestras manos, eso lo sabemos todos.

Si quereis encontrarme ya sabeis donde. Para cualquier duda, sugerencia, opinion o insulto que merezca la pena ser publicado:
blackngel_hack@hotmail.com

by blackngel

EOF

-[0x0B]----------
-[Rootkits]----------
-[by blackngel]----------SET-29--

blackngel_hack@hotmail.com

ROOTKITS

"Bienvenidos al mundo de lo real..."

&&&&&&--------& &&01&& INDICE & &&&&&&&&&&&&&

&&&&&&---------& &&02&& PROLOGO & &&&&&&&&&&&&&&

Como no... Otra vez mas me encuentro aqui con vosotr@s aportando algo mas de mis conocimientos, espero que valga la pena.

Cual es el plato del dia? mmmmm, pues hoy toca "rootkits" con tomate, tendreis que tragarme un rato pero tranquilos, este rollo de pasta no se hara muy largo.

Porque tengo que leer esto? Pues muy simple por lo mismo que aguanto
yo todos los dias la frase de un muy buen amigo en el messenger:

--Porque solo el poder del conocimiento nos hace libres by |_Tr0mP4s.

};.D

&&&&&&--------------&
&&03&& INTRODUCCION &
&&&&&&&&&&&&&&&&&&&&

Nuestras queridas rootkits tan conocidas actualmente no son tan
nuevas como parecen, segun parece datan de los aÏos 90 cuando aun
se estaban desarrolando muchos de los proyectos que ahora conocemos.

Estas herramientas son instalaciones que sustituyen a la mayoria
de las utilidades cruciales del sistema operativo y dejan
comprometida la maquina para futuras entradas, normalmente.

Uno ejemplo muy bueno de todo esto, es la tipica sustitucion del
binario "login" que se carga al inicio de cada sesion de un usuario
para autentificarlo como legitimo. El nuevo "login" que se instala
hace creer al usuario normal que la contraseÏa introducida es
incorrecta mientras la envia via e-mail al dueÏo de la rootkit el
cual a modificado dicho login para que contenga la direccion correcta.

Comencemos...

&&&&&&---------------&
&&04&& FUNCIONALIDAD &
&&&&&&&&&&&&&&&&&&&&

|==========="04.1-MANTENER ACCESO"==========|

Esto esta tipicamente muy relacionado con las backdoors (puertas
traseras), estas se encargan por ejemplo de ocultar aplicaciones
de acceso remoto.

Lista de metodos:

  --> Manejo del archivo inetd.conf para abrir conexiones.

  --> Ssh remote login, es un software utilizado por algunos
      atacantes amateurs.

  --> CGI Shell. Una rootkit que deja un script cgi hostil en el
      servidor web.

  --> Telnet Inverso. Me lo reservo para otro articulo.

  --> Magin packet-activated. Es una backdoor que abre un puerto

en la maquina, ejecuta un simple comando e inicializa una
sesion en el sistema.

--> SUID /bin/bash. Tipico... dejar escondida una copia de una
shell en otro directorio con permisos para cualquier usuario.

--> Muchos mas (Quiza en una actualizacion...)

El acceso local se mantiene con rootkits que tienen las herramientas
troyanizadas y necesarias que nos proporcionan un acceso con
privilegios de root (administrador).

Cabe destacar que mientras hay algunas rootkits que requieren de
nivel root para ser instaladas, hay otras que nos proporcionan un
metodo de escalada de privilegios.

|=========="04.2-EXPANSION TERRITORIAL"===========|

Simplemente decir que estas herramientas nos sirven para conseguir
acceso a otros sitemas, a esto se le suele llamar "ampliar el
territorio".

Estas contienen un sniffer que puede ser colocado en diversas
conexiones con protocolos en texto claro como pueden ser: telnet,
pop3, imap, ftp y algunos otros. Con esto conseguimos aparte
de muchos datos basura (o no tan basura), nombre de usuario y
passwords, que nos permiten acceso a la misma maquina o a otras
que son pertenecientes normalmente a la misma red (LAN).

Como todos sabemos los sniffers ponen un bit de la tarjeta de red
en "modo promiscuo", pues si mal no me acuerdo exite un modulo
cargable (LKM) que esconde tambien el estado de este bit. Solo
decir que esto lo consiguen desviando la syscall "sys_ioctl()" por
una alterada.
Cuando este metodo es implementado nos sirve de escondite para
cuando el administrador del sistema compruebe el estado de su
tarjeta red, ya sea mediante "ifconfig" u otras herramientas...
Mas adelante ya se hablara de los LKM.

Otro ejemplo de este tipo de herramientas es el SSH Syscall Sniffer
que actua interceptando llamadas al sistema como read() y write()
las cuales pueden contener datos beneficiosos para el atacante
(usernames y passwords).

Y ya para terminar este apartado tenemos tambien a keylogers
que tambien interceptan syscalls para controlar cada tecla pulsada
y reenviarla primero a un archivo determinado y permitir despues
su ejecucion normal para que el usuario continue con su trabajo
sin percibir ninguna anomalia en su sistema.

|=========="04.3-OCULTA EVIDENCIAS"========|

El tercer elemento crucial en la funcionalidad de una rootkit es
la eliminacion de evidencias. Esto se podria resumir en el borrado
de las huellas del ataque y la prevencion de nuevas huellas que
puedan quedar logeadas.

Todo esto se hace facil (hasta cierto punto) con el paso de zappers

para cada archivo de log como: utmp, wtmp, acct, messages, syslog, lastlog, loginlog.

Y tambien la modificacion del demonio syslogd que se encarga de todas las tareas de almacenamiento de informacion del usuario.

&&&&&&-------&
&&05&& TIPOS &
&&&&&&&&&&&&&

|==========="05.1-BINARY ROOTKITS"==========|

Este es el primer tipo de rootkits y los mas utilizados como se comento en la introduccion son los encargados de sustituir los binarios (ejecutables) mas importantes y ganar facilmente un acceso remoto o tambien local.

Los nuevos ejecutables estan especialmente troyanizados por el mismo programador y algunos se encargan de ocultar muchos de los procesos maliciosos a la vista del administrador o cualquier otro usuario.

Este tipo de rootkits suelen venir acompaÏadas de un script que se encarga de la instalacion (colocacion de cada binario en su correcto directorio). Todo esto siempre depende del SO al que estamos intentando comprometer, sobre todo porque suele variar la situacion de los archivos. No confundir con el sistema de archivos que por defecto siempre es "ext2" en las distribuciones linux.

Los binarios estan precompilados para cierta arquitectura como:
-Linux en un i386 INTEL
-Solaris en un Sparc

Aqui expongo una lista de los archivos que generalmente son los mas sustituidos por estas herramientas:

@==@
| amd   ftpd   netstat |
| asp   fusers   ntpd |
| basename   gpm   passwd |
| biff   grep   pidof |
| chfn   hdparm   pop2d |
| ch0sh   ident   pop3d |
| cron   ifconfig   ps |
| date   inetd   pstree |
| dirname   killall   rexed |
| du   login   rlogind |
| echo   ls   rcpinfo |
| egrep   lsof   rshd |
| env   mail   sendmail |
| find   mingetty   slogin |
| fingerd   named   sshd |
| su   telnetd   w |
| syslogd   timed   write |
| tar   top   wted |
| tcpd   traceroute   xinetd |

```
@============================================================@
```

Y aqui pongo las funcionalidades para las cuales se suelen
utilizar este tipo de paquetes:

   * Acceso remoto (Troyanizar "login" ...)
   * Acceso local (Troyanizar "login" ...)
   * Conexion ocuta (desapercivida) (Troyanizar "netstat")
   * Ocultacion de ficheros (Troyanizar "ls", "dir", etc)
   * Ocultacion de procesos (Troyanear "ps")
   * Ocultacion de las actividades del atacante (Troyanizar "syslogd")

Nota: Si quereis una explicacion mas detallada de cada uno de estos
      puntos, mail-me y estare encantado de explicarlo personalmente
      o realizar una actualizacion del articulo.

|=========="05.2-KERNEL ROOTKITS"==========|

Este segundo tipo de rootkits ya no son tan conocidos por este nombre
sino mas bien como Loadable Kernel Module (LKM, Modulos del Kernel Cargables),
pues si son modulos maliciosos que se pueden insertar en nuestro kernel
para que aporten nuevas funcionalidades (no muy amigables xD).

Estes modulos maliciosos normalmente se encargan de modificar las syscall
(system call, llamadas al sistema) que ya todos deberiamos de conocer.
Esto es un gran compromiso para el mismo sistema. Ya que intercepta cada
una de estas syscalls y produce una accion indebida, por ejemplo podria
capturar la syscall de la funcion read() (para leer un fichero) y realizar
otra operacion sobre el.

Si quieres la lista de las distintas "llamadas al sistema" las puedes
encontrar en "/usr/include/sys/syscall.h" o "/usr/include/syscall.h", el
codigo para cada call es parte del kernel.

Tambien cabe destacar que para que un LKM sea realmente bueno debe ser
capaz de esconderse hasta el mismo. Esto lo realiza con la alteracion de
la llamada al sistema "query_module()", para que cuando se ejecute el
comando "lsmod" el mismo no aparezca.

Bueno como esto ya escapa a las necesidades y aportaciones del articulo os
dejo unos buenos links para que continueis en este tema si os interesa:

--> http://www.giac.org/practical/gsec/Andrew_Jones_GSEC.pdf
--> http://rr.sans.org/threats/rootkits.php
--> http://hispakernel.rondanegocio.com/docs/intro_prog_drivers_linux.php
--> http://www.iespana.es/eMc2H/files/openbsd-lkm.txt (OpenBSD)

Nota: Buscar por "SucKit" un buen paquete sin duda alguna...

|=========="05.3-LIBRARY KITS"=============|

Los Library Troyan Kits, de los cuales T0rn 8 is el mas famoso, usan
diferentes metodos para eludir su deteccion. Por ejemplo, este kit, usa
una libreria de sistema especial llamada "libproc.a" que remplaza a una
libreria estandar la cual se encarga de la informacion de procesos
(sistema de archivos /proc).

Con esto conseguimos la ocultacion de ciertos procesos que a nosotros nos

interesen segun su nombre.
Estas rootkits solo suelen ser utilizadas para la ocultacion de procesos
y diversos archivos.

Nota: Mis conocimientos en este apartado son muy escasos tratare de
buscar mas informacion y ampliar este articulo. Os tendre
informados.

&&&&&&----------&
&&06&& SEGURIDAD &
&&&&&&&&&&&&&&&&&

|=========="06.1-DETECCION"============|

Existen algoritomos de "chequeo de suma" que nos sirven para
identificar ficheros legitimos de otros que no lo son.

Como ejemplo tenemos el MD5 que produce una cadena asegurandonos
de que la unica forma de que la cadena sea igual para los dos
ejecutables es que estos dos sean perfectamente identicos.
Un buen administrador deberia de guardar dichas cadenas de cada
ejecutable original en una fuente exterior, como puede ser en un
cd, una cinta magnetica, etc... Asi, podria utilizarlas despues
para asegurarse de la normalidad en su sistema.

Una gran utilidad que cumple todas estas funciones es "TripWire"
que controla la integridad de cada archivo del SO. Esta
herramienta se encuentra disponible para sistemas Unix/Linux en:
http://www.tripwire.org.

Tampoco nos podemos olvidar de realizar escaneos desde otras
maquinas a la nuestra con el objetivo de encontrar puertos abiertos
utilizados por backdoors. Puertos que normalmente deberian de
estar cerrados segun nuestro criterio.

Tambien tenemos a "Chkrootkit" que funciona en busca de patrones
de rootkits conocidas. Esto lo realiza buscando archivos
caracteristicos o cadenas dentro de ejecutables que son sabidas
de antemano.

Y por ultimo tenemos demonios como rkdet, que se encargan de
detectar cualquier intento de instalacion de una rootkit, y si
es posible, evitarlo y avisar de alguna manera al mismo administrador.

|=========="06.2-PROTECCION"===========|

Para que se consiga instalar un rootkit en nuestro sistema,
evidentemente es logico que primero haya que penetrarlo, entonces
no tenemos mas solucion que proteger nuestro equipo, tanto fisicamente
como todo el entorno de red en el que trabaje.

Ya saben, cierren servicios no utilizados o por defecto, instalense
un antivurus actualizado, un firewall perfectamente configurado, si
hace falta que lo haga una persona con mas conocimientos mejor y por
ultimo miren haber a quien dejan tocar su ordenador y que tipo de

software ejectuan en el.

&&&&&&------------&
&&07&& USO NORMAL &
&&&&&&&&&&&&&&&&&&

Bueno, pues ya para terminar, aqui explicare el uso normal que se suele
dar a estas herramientas tan utiles.
Solo seran unos pasos generales que la mayoria ya deberiais de saber
pero que no viene mal recordar...

**PASOS PRINCIPALES:

1.--> Busca un host vulnerable, ayudate de un scanner automatico
de alguna vulneravilidad conocida. (Simplemente arreglatelas).

2.--> Explota el sistema, curratelo y conviertete en root.

3.--> Descargate una "Rootkit" (Segun tu preferencia). Asegurate de
que sea compatible con el sistema irrumpido.

4.--> Instala la herramienta en el sistema, suelten traer un script
que hara todo el trabajo por ti.

**BORRAR HUELLAS:

5.--> Deshabilita el Shell History (unset HISTFILE;
export HISTFILE=/dev/null).

6.--> Mata el daemon syslog y libera los logs del sistema. (Kill).

7.--> No te olvides de pasar algun zapper. (Cloak, Zap2, etc).

8.--> Elimina cualquier otra evidencia o rastro que haya podido
quedar por la instalacion del rootkit u otros ficheros.

&&&&&&----------------------&
&&08&& ROOTKITS EN WINDOWS &
&&&&&&&&&&&&&&&&&&&&&&&&&&&

|==========="08.1-INTRODUCCION"==========|

Si bien ya estamos hartos de ver rootkits para nuestros queridos SOs
"Unix/Linux", pues, en 1999, tambien se diseÏo uno para que trabajara en la
plataforma NT de Windows.

Este se basaba en una tecnica llamada "enganche de funciones" que en lenguaje
tecnico se trata de apoderarse de las llamadas al API que como muchos ya

sabremos (Los crackers y viruckers mas...) son el motor principal de nuestro software ya que en ultima instancia siempre estan dependiendo de ellas.

Con todo esto conseguimos mas o menos los mismos objetivos que en nuestras plataformas *nix. Ocultacion de archivos, procesos y redireccion de ejecutables.

|=========="08.2-DETECCION"=============|

Si el rootkit esta desarrollado realmente a conciencia sera casi imposible el detectarlo en nuestro propio sistema, ya que como dije antes, el sistema siempre esta dependiendo del API y las llamadas a esta estarian troyanizadas (cambiadas por otras con diferentes intenciones ;)).

Nuestro punto mas fuerte de deteccion sera la busqueda de backdoors que haya dejado o servicios prestados al exterior de cuya utilidad desconocemos.
Como no, tendremos que hacer un escaneo de puertos desde fuera y si es posible (siempre) realizar una auditoria del sistema que haria de por si el portscan y buscaria mas vulneravilidades o errores desconocidos.

Desde el interior deberiamos buscar tambien cambios en las entradas del registro del sistema.

||||||=08.2.1=||||||=Rootkit Detector=||||||

Esta es una herramienta estupenda que se encarga de la deteccion de los rootkits mas conocidos hasta el momento, se vasa en la busqueda de patrones. Funciona bajo entornos NT. (En XP tambien xD)

Aqui os pongo una salida del programa en mi sistema (no encontro ninguna por supuesto ;D):

```
_____________________________________________________________________
        . .. ...: Hax0rcitos Rootkit Detector v0.3b :... .. .       |
Rkd v0.3b - Rootkit Detector                                        |
Programmed by aT4r@3wdesign.es                                      |
Copyright (c) 2003 3W Design, Security                              |
http://www.3WDesign.es                                              |
                                                                    |
                                                                    |
-Gathering Service list Information... ( Found: 248 services )      |
-Gathering process List Information... ( Found: 27 process )        |
-Searching for Hidden process Handles. ( Found: 0 Hidden Process )  |
-Searching again for Hidden Services.. ( Found: 0 Hidden Services)  |
-Searching for wrong Service Paths.... ( Found: 0 wrong Services )  |
-Searching for Rootkit Modules........ ( Found: 0 Suspicious modules )  |
_____________________________________________________________________|
```

* Con el Panda Antivirus instalado, "Titanium" segun mis pruebas, marca otra salida, en este caso saldria 1 en wrong Service Paths y una mencion sobre un archivo llamado "pavsrv.exe", no se que funcion realiza este programa situado en ’system32’ tratare de averiguarlo pero en tal caso deberia ser un proceso oculto. Si alguien sabe algo o quiere ayudarme a saber algo sobre esto lo agradeceria mucho.
  O sera que no deberian ejecutarse servicios desde este directorio?

Algunas de las rootkits detectadas:

* AFX Rootkit 2003

```
* Hacker Defender.
* He4HookInv.
* Rkit (Backdoor.W32/ASW)
* NT Rootkit _root_
```

Como otra caracteristica, cabe destacar que RKD almacena una base de datos interna con las firmas MD5 de algunas rootkits, exploits y programas de hack conocidos, esto reducira aun mas los posibles problemas de penetracion.

USO:

```
   rkd.exe [parametros]
        -v              - Muestra informacion extra.
        -m              - Muestra el md5 de un fichero.
```

*Realmente dificil eh! ;)

|=========="08.3-LOGS GENERADOS"========|

Realmente solo se generaran logs de importancia si el rootkit contiene un fallo en su programacion, sino, unicamente nos podremos encontrar como mucho, errores en la incapacidad de arrancar algun servicio del sistema que se veran reflejados en los archivos de logs habituales de nuestro SO.

```
&&&&&&-----------&
&&09&& DESPEDIDA &
&&&&&&&&&&&&&&&&&
```

Chic@s esto ha sido todo por hoy, creo que basta para tener una idea de los mas generalizada sobre lo que pueden y son capaces de hacer estas grandes herramientas.

Mas informacion sigue esperando ahi fuera para ser leida por vosotros, ya no tendriais que estar leyendo esta despedida y si tendriais que estar aprendiendo nuevos contenidos del mundo informatico.

Como consejo os diria que buscaseis textos sobre programacion de LKM’s que realmente los hay muy buenos. Me gustan las traducciones de la "phrack" y mucho ha sido escrito ya. Es un tema muy bonito y se pueden llegar a hacer maravillas, hay que tener en cuenta que estamos trabajando a nivel del kernel, el maximo privilegio!! alguien da mas?

Opinion: Creo que es lo mas inteligente que podemos hacer despues de comprometer un sistema. Hoy por hoy, aquellos trucos antiguos que todos tanto conocemos nos meterian en el trullo.

Importante: Me han comentado que andan por ahi sueltos unos cuantos IDS, tener cuidado con ellos que son unos chivatos xD.

Cualquier duda, opinion, complemento a este articulo o insulto (que valga la pena) ya sabeis donde encontrarme: blackngel_hack@hotmail.com.

by blackngel

EOF

-[0x0C]--
-[3er articulo publicado por SET en @rroba]------------------------------
-[by SET Ezine]---SET-29--

El presente articulo fue escrito a inicios del 2001. En este lapso de tiempo
ha llovido bastante y en el mundo de la informativo mucho mas.
No espereis encontrar aqui grandes descubrimientos !

INTERNET DESDE UNA UNIVERSIDAD

Los acontecimientos aqui descritos se refieren a personas y situaciones
ficticias, pero que son muestra de miriadas de hechos ocurridos y que siguen
ocurriendo en las redes de habla hispanica (...y estamos seguros que estos
ejemplos se puede extender al resto del mundo).

1.-INTRODUCCION

La historia que os queremos contar entra dentro de la normalidad de un pais
que nos empenyamos en llamar Espanya cuyas Universidades son una hirvierte
cazerola donde se mezclan tecnologias punta, hardware de hace diez anyos,
gente adormilada, zanganos contumaces, personas con ganas de renovar y
renovarse, todo ello con algunos ingredientes de endogamia e irresponsabilidad.

2.-TENEMOS UN NUEVO BECARIO

Probablemente no lo sabeis, pero en las universidades de Spain, el numero de
becarios es casi igual al de funcionarios de contrato fijo. Esto no tiene nada
de malo, recordad que estamos hablando de un pais de lengua NO anglo-sajona y
sin ninguna similitud con otro pais de gran poderio y con presidentes de
aficiones extravagantes, siempre que se respeten las reglas del juego.

Y cuales son estas reglas ?, Se preguntaran nuestros lectores.... pues dificil
es el conocerlas. Becarios hay muchos, pero casi ninguno de ellos conoce el
reglamento bajo el cual se rige su vida laboral. Si alguno de vosotros no dais
credito a lo que decimos, no teneis mas que acercaros a la ventanilla de una
universidad y preguntar por semejante reglamento. En caso de que tengais algun
conocido que trabaje como becario, le pedis que realice esta gestion por
vosotros y os aseguramos que obtendreis el mismo resultado... ninguno.

Pero al equipo de SET no se le escapa nada de lo que ocurre o pueda ocurrir en
nuestro entorno y ha obtenido esta informacion para todos vosotros ! Todo el
cumulo de papeles que hemos logrado se pueden resumir en tres pequenyas reglas:

- Un becario debe estar supervisado por un funcionario de contrato fijo.
- El trabajo de un becario no debe remplazar al de un funcionario fijo.
- Un becario no puede realizar el mismo trabajo, mas de dos anyos.

Estas reglas son de una unica universidad, pero en general todas siguen los
mismos parametros.

Aparentemente son reglas faciles de aplicar y de contenido mas que razonable
pero si continuais leyendo, entendereis porque esta informacion esta tan
celosamente custodiada.

Una manyana cualquiera en un departamente como muchos en una de tantas unis...

'Hola,...soy Josefina Trabajo-Adicta, he sido contratada como becaria y me
comunicaron que viniera a presentarme aqui'

’Buenas dias !,....soy Pepito Explotado, tu primer dia, eh?, aqui esta Felix El-Escaqueado y aquella es Maruja Metementodo’

Nuestra flamante becaria, observa la edad media de los presentes y se pregunta extranyada como puede ser que todos los funcionarios sean tan jovenes. Inocentemente deja caer.

’Asi que tu eres el jefe de este departamento?’

La respuesta es fulminante.

’Noooo, aqui la jefa es Flora Nopegagolpe, ... bueno la jefa y la unica con contrato fijo, TODOS los demas somos becarios’

.....os acordais de uno de los puntos de las reglas fijas e inamovibles de los becarios ? Esta es una situacion general, las universidades, con el animo de captar nuevos alumnos (la baja natalidad ha hecho estragos entre los ’clientes’ de estos entes), han creado nuevos departamentos y servicios, pero no disponen de personal suficiente. La solucion ha sido hacer mal uso de la figura del becario, deformando su perfil. De ser una persona que va ha realizar una tarea unica y creativa, pasa a ser un simple administrativo.

3.-EMPEZAMOS A TRABAJAR

El trabajo en cuestion, que en teoria habia sido vendido como creativo y formador no era otra cosa que un normalillo toma-datos de la gente que venia por ahi y hace-fotocopias para otros departamentos (que probablemente se apresuraban a tirar a la papelera al recibirlos).

El entorno era agradable, unas oficinas bien instaladas con equipos informaticos para todos los gustos y una red de acceso local con conexion hacia el exterior. Los siguientes factores contribuian a que en dicha oficina se empezaran a formarse grupos, foros y demas reuniones de gentes diversas en funcion de las horas del dia :

- La Sra. Nopegagolpe, hacia honor a su nombre y siempre estaba fuera de la oficina en reuniones de enorme importancia y dudosa utilidad.
- El trabajo no desgastaba las meninjes a nadie.
- Los equipos informaticos eran de primera fila.
- La conexion a Internet de primer orden.
- ....aire acondicionado.

Nuestra Trabajo-Adicta, pronto se dio cuenta de que el asunto del trabajo era peripatetico. Lo normal en aquel centro eran reuniones a la hora del cafe, gente que venia a utilizar los equipos, porque los de la biblioteca estaban saturados y habian oido que ahi tenian buen acceso, novias de amigos de becarios que venian a pasar e-mails particulares, amigos de novias que querian bajarse el ultimo MP3 y deseaban buena velocidad de bajada y asi un largo etc.

Podemos resumir el exito de audiencia ante los terminales del departamento en dos :

- Realmente la velocidad de conexion al exterior era muy buena.
- Habia poco que hacer.

Las razones de la bondad del servicio reside en el acuerdo que hay establecida en este pais para las conexiones de los organismos dedicados a la ensenyanza. Realmente quien realiza la conexion es la REDIRIS, utilizando algo realmente fuerte con capacidades superiores a las 34/155 Mbps. Si alguien desde dentro de la universidad consigue una conexion sin filtrar, tiene a su alcance una velocidad que para si quisieran muchos de los proveedores de Internet. Evidentemente

todos los zanganos del lugar, habian descubierto esta posibilidad y ahi estaba la explicacion de semejante desfile de desocupados.

El segundo punto se debia a que la Sra. Nopegagolpe era sumamente habil en inflar las tareas asociadas a su departamente frente sus superiores. Estas tareas se resumian en registrar a los alumnos que deseaban acceder a determinados servicios gratuitos pero que requerian un cierto control.

Lo que si era cierto, era la estacionalidad del trabajo. Normalmente habia agobio al principio de cada periodo lectivo y despues tranquilidad absoluta. Trabajo-Adicta, no estaba en absoluto de acuerdo con la situacion pero poco podia hacer salvo ingeniarse para que una propuesta suya saliese hacia las altas esferas de las jefaturas.

La propuesta consistia en crear una web desde la cual los mismos estudiantes pudieran darse de alta en los diversos servicios. A partir de ahi todo el trabajo administrativo posterior se limitaba a comprobar que los datos eran correctos (realmente el Sr. Nosabenada, estaba matriculado?) y se eliminaba todo el trabajo posterior de copias y envios por correo.

A pesar de que a Nopegagolpe, no le hizo ninguna gracia (ya que limitaba sus futuras intenciones de pedir mas becarios, alegando trauma funcional), la idea cuajo inmediatamente y se decidio implementarla rapidamente.

4.-....Y SI TODOS SE REGISTRARAN POR LA RED ?

La primera idea se limitaba a crear el acceso localmente desde una maquina que estuviera en red con el servidor del departamento (un NT que hacia al mismo tiempo de workstation de una becario,....entre nosotros, un verdadero desastre desde el punto de vista de la seguridad) pero dada la moda de impulso de iniciativas y modernidad, y la falta de verdaderas ideas, la novedad se puso como ejemplo y a alguien en las altas esferas con muy pocas luces pero con enormes ganas de ponerse medallas, decidio que dar nuevo impulso a la iniciativa, modificandola de la forma siguiente :

- Los accesos se pudieran hacer sin necesidad de personarse en la oficina (o sea desde cualquier punto de Internet)
- El propio departamento fuera el encargado de implementarla.

A cualquiera que tenga un poco de sentido comun, no se le escapa que un departamento donde se permite que un servidor de red, sea simultaneamente utilizado como estacion de trabajo administrativo, no era el mas adecuado para encargarse de semejante labor, ya que no parecian tener criterios muy claros acerca de la seguridad digital.

Ademas, conociendo la composicion de las fuerzas vivas de dicho departamento, (solo Nopegogolpe era funcionaria con contrato fijo), era evidente que serian los propios becarios los que se iban a encargar de la tarea. Tampoco dichos heroes se iban a preocupar mucho de la seguridad (total ! Al cabo de dos anyos, a la calle !).

5.-PRUEBAS Y DESASTRES

Asi armados, con mas buena voluntad que buen juicio, Trabajo-Adicta y sus colegas se lanzaron a la tarea sin ninguno de los pasos que se requieren en la implantacion de cualquier proyecto de este tipo. Por si el avispado lector no se ha dado cuenta, en el fondo lo que se estaba pidiendo era :

- Acceder a una base datos desde cualquier sitio del mundo.
- Poder actualizar los registros.

- Reutilizar un servidor existente que se encontraba dentro de una organizacion con muchas subredes.

Los minimos pasos a seguir cuando alguien se encuentra con una tarea de tales dimensiones, son :

- Definir el entorno de los que pueden acceder a la aplicacion.
- Definir el entorno de la base de datos y soft asociado.
- Definir el soft que debia gestionar el servidor.
- Realizar un minimo estudio de las consecuencias si, a pesar de las precuaciones tomadas, alguien conseguia acceso no autorizado.

Nada de todo esto se hizo. Nuestros becarios, simplemente se esforzaron en que la aplicacion funcionara (fundamentalmente, se ocuparon de que la presentacion que Nopegogolpe debia realizar ... dentro de quince dias !, fuera lo mas bonita posible). Para ello siguieron los pasos siguientes :

- Buscaron en Internet un servidor de PHP de libre acceso (gratuito)
- Se leyeron en unas horas las instrucciones
- Instalaron y configuraron en un dia las bases de datos

...y lo que no hicieron :

- No comprobaron la configuracion del servidor NT que debia alojar la base de datos.
- No limitaron el acceso fisico a dicho servidor.
- No se leyeron (con calma) la documentacion que se bajaron sobre PHP.

Esto dentro de lo mas gordo, pero tampoco pretendemos escribir un documento sobre seguridad......

6.-DESENLACE

Pues sucedio lo que cualquiera se podia imaginar. Los scripts escritos en PHP son famosos por dos cosas, la rapidez con que se escriben y la facilidad con que dejan agujeros de seguridad. (Interesados, pueden consultar la web http://oliver.efri.hr/~crv/security/bugs/mUNIXes/http13.html)

Al no leerse con calma la documentacion, dejaron en el mismo directorio de acceso libre, un scrip en PHP que permitia leer y cambiar la password de administracion. Alguien lo utilizo para descubrir la password de acceso.

Como ademas, en la pagina principal, decian quienes eran y donde estaban, busco el servidor y encontro las tipicas agujeros de las instalaciones por defecto de un NT de hace dos anyos. Armado con un poco de paciencia, filosofia, ingenieria social y la password anteriormente citada, tomo control del servidor.

En dicho servidor, el administrador del dominio principal se habia dejado su password (el archivo SAM de las maquinas que corren NT, es una autentico fuente de alegrias para los hackers de este planeta). A partir de ahi, nuestro anonimo visitante hubiera podido hacer lo que hubiera deseado, pero siempre hay un angel protector para inocentes e inexpertos.

Entre la pleyade de visitantes (fisicos) que frecuentaban aquel departamento (y que jamas hubieran debido hacerlo), habia uno que ademas de leerse el correo y bajarse musica MP3, sabia algo de sistemas y tenia un poco de curiosidad. Dado que el servidor estaba siempre conectado con pass de administrador, pudo ver que alguien habia hecho el dia anterior una copia de seguridad de la SAM. Esto le extranyo sobremanera (tal celeridad en aquel departamento era inpensable o como minimo inesperada) y se lo comunico a la novia del amigo de uno de los becarios. Como las malas noticias corren deprisa, nuestros becarios

leyeron un tocho de documentacion en unas horas y descubrieron sus monumentales errores. Rapidamente se lo comunicaron al administrador del dominio (nada de e-mails, faxs ni nada que dejara rastro,....lo fueron a ver en persona).

Como todo el mundo estaba implicado, no se dio noticia del hecho, solo se cambiaron las passwords y se modificaron los scripts.....nadie llego a saber quien fue el anonimo visitante, ni lo que realmente hizo, ni se comprobo,...si todavia seguia alli.

****
*********LAS CONEXIONES REDIRIS SEGUN UN ESTUDIANTE*********
****

La mayor parte de los organismos oficiales y entes sin animo de lucro en Espanya, utilizan los servicios de la REDIRIS para sus conexiones telematicas. Como ejemplo, ahi va una lista de los entes asociados en Andalaucia.

CAHA Observatorio Hispano-Aleman ’Calar Alto’
     Instituto Max Plank/Comision Nacional Espanyola de Astronomia
CARTUJA.CSIC Centro de Investigaciones Cientificas Isla de la Cartuja
CAP.JUNTA-ANDALUCIA Centro Andaluz de Prospectiva
CEC.JUNTA-ANDALUCIA Consejeria de Educacion y Ciencia
CDMA.JUNTA-ANDALUCIA Centro de Documentacion Musical de Andalucia
CICA Centro Informatico Cientifico de Andalucia
CID.CAP.JUNTA-ANDALUCIA Centro de Investigacion y Desarrollo
      Consejeria de Agricultura Y Pesca
CMA.JUNTA-ANDALUCIA Consejeria de Medio Ambiente
CNA Centro Nacional de Aceleradores
CPRECAAN.JUNTA-ANDALUCÖA Consejeria de Presidencia de la Junta de Andalucia
EAMS.FUNDEA Euro Arab Management School
EASP Escuela Andaluza de Salud Publica
EBD.CSIC Estacion Biologica Donyana
EEA.CSIC Escuela de Estudios Arabes
EEHA.CSIC Escuela de Estudios Hispano Americanos
EELM.CSIC Estacion Experimental La Mayora
EEZ.CSIC Estacion Experimental del Zaidin
EEZA.CSIC Estacion Experimental de Zonas Aridas
FEM Fundaci¢n de Estudios Marinos
HU.SAS.JUNTA-ANDALUCIA Hospitales Universitarios Andalucia
IAA Instituto de Astrofisica de Andalucia
IACT.CSIC Instituto Andaluz de Ciencias de la Tierra, CISC/UGR
IAPH.JUNTA-ANDALUCIA Instituto Andaluz de Patrimonio Historico
IAS.CSIC Instituto de Agricultura Sostenible
IAT Instituto Andaluz de Tecnologia
IBVF.CARTUJA.CSIC Instituto de Bioquimica Vegetal y Fotosintesis
ICMAN.CSIC Instituto de Ciencias Marinas de Andalucia
ICMSE.CARTUJA.CSIC Instituto de Ciencias de Materiales de Sevilla
IEA.JUNTA-ANDALUCIA Instituto de Estad¡stica de Andalucia
IEO-CADIZ.IEO Instituto Espanyol de Oceanografia - Cadiz
IEO-MALAGA.IEO Instituto Espanyol de Oceanografia - Malaga
IESAA.CSIC Instituto de Estudios Sociales Avanzados de Andalucia
IG.CSIC Instituto de la Grasa y Derivados
IGEO.CSIC Instituto de Geologia - Granada
IIQ.CARTUJA.CSIC Instituto de Investigaciones Quimicas
IMSE.CNM Instituto de Microelectronica de Sevilla
IPB.CSIC Instituto de Parasitologia y Biomedicina ’Lopez Neyra’
IRM.IGN Instituto de Radiometria Milimetrica
        Instittuo Max Plank/CNRS/Instituto Geografico Nacional
IRNASE.CSIC Instituto de Recursos Naturales y Agrobiologia
JRC Instituto Europeo de Prospectiva Tecnologica ’JRC’
    (Comision Europea W.T.C.)

```
OSN Observatorio de Sierra Nevada
PARLAMENTO-AND Parlamento de Andalucia
PSA Plataforma Solar de Almeria
ROA Real Observatorio de La Armada de San Fernando
SAS.JUNTA-ANDALUCIA Servicio Andaluz de Salud
SBA.JUNTA-ANDALUCIA Sistema Bibliotecario Andalucia
SICMA.CMA.JUNTA-ANDALUCÖA Servicios de Investigacion de Consejeria de Medio
                          Ambiente
UIA Universidad Internacional de Andalucia, sede Sevilla
UNIARA.UIA Universidad Internacional de Andalucia
           Sede Iberoamericana Santa Maria de La Rabida
UNIAAM.UIA Universidad Internacional de Andalucia, Sede Antonio Machado
UIDA Instituto Andaluz del Deporte
UNA.CSIC Unidad de Nutricion Animal
UALM Universidad de Almeria
UCA Universidad de Cadiz
UCO Universidad de Cordoba
UGR Universidad de Granada
UHU Universidad de Huelva
UJAEN Universidad de Jaen
UMA Universidad de Malaga
US Universidad de Sevilla
UPO Universidad Pablo de Olavide

Todo ello esta gestionado desde una red que parte de Madrid

Dominio: rediris.es
Estado: Delegado
Organizaci¢n: RedIRIS
Organizaci¢n: Centro de Comunicaciones CSIC RedIRIS
Direcci¢n (Calle,No...): Serrano 142
Direcci¢n (Municipio): Madrid
Direcci¢n (Cod. Postal): E-28006
Direcci¢n (Provincia): MADRID
Direcci¢n (Pais): SPAIN
C. Administrativo: VC63
C. Tcnico: MC34-ESNIC
C. Facturaci¢n: AMD1-ESNIC
Serv. primario: sun.rediris.es 130.206.1.2
Serv. secundario: chico.rediris.es 130.206.1.3
Serv. secundario: tais.rediris.es 130.206.1.39
Acronimo de proveedor: REDIRIS

La direccion IP nos indica que es una red de Clase B, con una capacidad de dar
servicio a unas posibles 65025 maquinas sin contar que algunas de ellas pueden
dar a su vez servicio a pequenyas redes locales.

Los routers por los que pasan estos chorros de informacion son del tipo AS5200
, Cisco 2501/5260/5300 terminal server IOS 11.3.6(T1) o Cisco IOS 11.3 -
12.0(9).

En general la seguridad del conjunto es bastante robusta, fueron pioneros en la
utilizacion de PGP y normalmente recomiendan software GNU. Como ejemplo de todo
esto basta con intentar ejecutar un escaneo de puertos sobre cualquiera de las
maquinas principales y vereis que la respuesta es bastante brusca.

Actualmente se encuentran colaborando en el grupo de trabajo de Internet2, para
probar redes de muy alta velocidad cuya aplicacion inmediata seria las
transmisiones de eventos medicos (operaciones en tiempo real), multiconferencias
realmente operativas (si alguno a participado en alguna multiconferencia de
las 'clasicas' sabra a que me refiero), etc.
```

Todo este conjunto de herramientas, hacen las delicias de los aprendices de hackers en las Universidades. No por sus ataques a los recursos de la REDIRIS, sino por su utilizacion. Si alguno de ellos consigue una conexion de primera fila sin control del ancho de banda, la unica limitacion que tendra para bajarse todo tipo de soft, musica o peliculas, sera su capacidad de almacenamiento local.

****
*********LOS PELIGROS DE LOS CGI**********
****

Como evitar y porque, los metacaracteres en los CGI Scripts que se ofrecen libremente en la red.

En la red podemos encontrar miles, cuando no millones de CGi que hacen casi de todo. Cuando alguien recibe el tipico encargo/marron de ultima hora, es muy frecuente que se utilicen este tipo de software de libre distribucion. El problema reside en que no todo el software bajo GNU es fiable ni siquiera decente. En el caso de los CGIs, sean en perl, C,.... siempre se deben comprobar dos cosas fundamentales :

- Que no permitan ejecutar comandos arbitrarios bajo inputs banales.
- Que no permitan la ejecucion de archivos arbitrarios

Como impedir ejecucion de comandos arbitrarios.

En el caso de un script CGI, que admita datos del usuario, no importa el origen pero supongamos que se inyectan desde la variable de sistema $QUERRY_STRING. Es muy importante la forma en que se 'desinfectan' los datos que nos llegan desde esta variable para impedir sorpresas desagradables.

Ejemplo en Perl, a evitar (se controlan los caracteres conocidos que se desean impedir su ejecucion) :

```
#!/usr/local/bin/perl
$datos_usuario = $ENV{'QUERY_STRING'}; # Se aceptan los datos
print "$datos_usuario\n";
$datos_usuario =~ s/[\/ ;\[\]\<\>&\t]/_/g; # ERROR. Se eliminan algunas
                                           #        caracteres
print "$datos_usuario\n";
exit(0);
```

Actuando de esta forma, se eliminan algunas caracteres, pero hay que tener en cuenta que la imaginacion de las personas es ilimitada y siempre encontraremos a alguien que se le ocurren caracteres diabolicos.

Mucho mejor es dejar pasar solo los caracteres deseados:

El CGI en perl del ejemplo anterior, es mucho mas seguro de la forma siguiente:

```
#!/usr/local/bin/perl
$_ = $datos_usuario = $ENV{'QUERY_STRING'}; # Se aceptan los datos
print "$datos_usuario\n";
$OK_CHARS='-a-zA-Z0-9_.@';     # Lista de caracteres aceptable
                               # Esta lista la podemos modificar en funcion
                               # del ultimo RFC que nos interese
s/[^$OK_CHARS]/_/go;
$datos_usuario = $_;
print "$datos_usuario\n";
exit(0);
```

La misma tecnica podemos (y debemos) emplear cuando queramos ejecutar un archivo. Siempre debemos partir de una lista de archivos autorizados, sino corremos el riesgo de ejecutar el ultimo back-door que nos copiaron en el disco utilizando otros caminos, en lugar de nuestro flamante programa de actualizacion de bases de datos.

Como normas generales, se den borrar todos los scripts que no sepamos para que sirven y los de demostracion.

Si hay alguien que quiera mas informacion, aqui les damos algunas direciones, aunque nuestra experiencia no es tanta de falta de informacion como de exceso de desidia en leer (e intentar entender) la disponible.

Documentos generales de advertencia.

CERT Advisory CA-97.12 "Vulnerability in webdist.cgi"
AUSCERT Advisory AA-97.14, "SGI IRIX webdist.cgi Vulnerability."

WEBS.

http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html
http://www.genome.wi.mit.edu/ftp/pub/software/WWW/cgi_docs.html

LIBROS:

1.-Practical Unix & Internet Security, Simson Garfinkel and Gene Spafford, 2nd edition, O’Reilly and Associates, 1996.
2.-Programming Perl, Larry Wall, Tom Christiansen and Randall L. Schwartz, 2nd edition, O’Reilly and Associates, 1996.

EOF

```
-[ 0x0D ]-------------------------------------------------------------------
-[ Ensamblador orientado al cracking ]--------------------------------------
-[ by ThEnemI ]-------------------------------------------------------SET-29--
```

ENSAMBLADOR ORIENTADO AL CRACKING

by ThEnemI
<thenemi@hotmail.com>

ADVERTENCIA

El autor no se hace responsable del mal uso que se le pueda asignar al contenido de este texto. Esto se hace con fines educativos. Ademas, no he sido yo, nadie me ha visto, y no teneis pruebas. xD

Haz lo que te de la gana con el texto

INDICE

- practicar

0x09 Despedida y Reflexion

0x00. Introduccion

Este texto no pretende ser una guia avanzada, ni tan siquiera una guia. Es simplemente una "recopilacion" de textos LIBRES que vagan por la red. Si ya sabes lo basico, deja este texto y coge algo mas complejo. ;>
Pero si antes de leer textos sobre como crackear x crackme te gustaria saber algo de ensamblador... Este es tu articulo

0x01. FAQ (Frecuently asked questions)

Se que a muchos de vosotros esto os parecera aburrido, y quereis pasar enseguida a la accion (al ataquerrrr!!!) PERO.... Si no sabemos que es lo que estamos haciendo...
?:/ (no coment). Bueno, ahi van....

/// Que es el cracking?
Es una tecnica que consiste en la desproteccion de SOFTWARE. Ojo, no confundir con una tecnica de hacking que se llama igual, y es mas bien destrozar todo lo que pilles (desconectando ordenadores de Inet, jo$%!@ el HD*....)
*HD: Hard drive (disco duro)

/// Que es ensamblador?
Ensamblador es un lenguaje de programacion de bajo nivel, con las consecuencias que ello conlleva:
1. Es dificil de aprender (ya con cara de panico? tranquilo, solo te costara al principio)
2. Exige conocer todos los componentes del ordenador (auch!)
3. Obtenemos el maximo rendimiento!!
De momento no os preocupeis, para el cracking solo necesitareis conocer las ordenes basicas, aunque os recomiendo enteraros un poquillo mas...

/// Que herramientas voy a necesitar?
Pues para aprender ensamblador, lapiz, papel, interes, y un debug (dios, no!!!!) tranquilo, si usas windows* (XP, ME, 9x*,NT...)Viene incluido. Dependiendo de la version, en accesorios, MS-DOS*,o en accesorios, simbolo del sistema, o en "ejecutar" escribe "command.com"

********Comenario SET
Me parece que command.com era en windows 3.11, apartir de Win-95 y desde luego XP, 2000, NT se tiene que escribir "cmd"
********Fin

*windows: Es un sistema operativo. Es decir, el que interpreta las instrucciones. Ya es hora de que sepas que hay mas sistemas operativos, como puede ser MS-DOS, Linux y sus versiones (Suse, RedHat...), y la familia UNIX, SunOS, Solaris...

Para crackear: pues el progama objetivo, y un debugger mas potente (olly, softice...) y un editor hexadecimal. De momento, no os preocupeis.

*M$-DO$: Microsoft Disk Operative $ystem. El primer $istema Operativo (O$ en ingles)de la queridisima compaÏia de Billy.

/// Hexadecimal
Pues supongo que todos han oido hablar del sistema decimal (10 digitos, del 0 al 9) o del sistema binario (con 0 y 1) Pues HEXA, no es mas que un sistema, en que se usan numeros del 0 al 9, y letras de la A a la F.[0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F] Para que quede claro: 1=1; 2=2; 3=3; 4=4; 5=5; 6=6; 7=7; 8=8; 9=9; A=10; B=11; C=12; D=13; E=14; F=15.

/// Yo valgo?
Pues si tienes interes, y usas un poquito lo que hay debajo del pelo... (eso no, la cabeza!)

/// Tengo un problema...
Para cualquier duda, escribe a <thenemi@hotmail.com>

0x02. El ordenador por dentro
Pues para conocer tu ordenador (un vistazo rapido), que mejor que unos apuntes de la Escuela Politecnica Superior de Alicante, y otros de la universidad de Guadalajara. Digo yo que tendran que saberlo... (se pueden encontrar navegando por ahi, como ya dije esto tiene mucho de recopilacion)

A. Que es un ordenador?
Un ordenador es una maquina para el tratamiento de informacion, que funciona manipulando seÏales electricas binarias (0 y 1) y que se constituye de los siguientes elementos:

[ELEMENTOS DEL SISTEMA DE COMPUTO]
Le llamamos sistema de computo a la configuracion completa de una computadora, incluyendo las unidades perifericas y la programacion de sistemas que la hacen un aparato util y funcional para un fin determinado. Se compone de los siguientes elementos:

* La memoria (RAM/ROM/CMOS/EPROM/CACHEs)

Es un conjunto de celdas (actualmente fabricadas con semiconductores) usadas para procesos generales, tales como la ejecucion de programas y el almacenamiento de informacion para las operaciones.

Cada una de las celdas puede contener un valor numerico y tienen la propiedad de ser direccionables, esto es, que se pueden distinguir una de otra por medio de un numero unico o direccion para cada celda.

El nombre genrico de estas memorias es Random Access Memory (Memoria de acceso aleatorio) o RAM por sus siglas en ingles. La principal desventaja de este tipo de memoria es que los circuitos integrados pierden la informacion que tienen almacenada cuando se interrumpe la alimentacion electrica. Esto lleva a la creacion de memorias cuya informacion no se pierda cuando se apaga el sistema. Estas memorias reciben el nombre de Read Only Memory (Memoria de

solo lectura) o ROM.

* Las unidades de entrada y salida

Para que una computadora nos sea util es necesario que el procesador se comunique al exterior por medio de interfaces que permiten la entrada y la salida de datos del procesador y la memoria. Haciendo uso de estas comunicaciones es posible introducir datos para su procesamiento y la posterior visualizaci¢n de los datos ya procesados.

Algunas de las unidades de entrada mas comunes son teclados, lectoras de tarjetas (ya en desuso), raton, etc. Las unidades de salida mas comunes son las terminales de video(monitor...) y las impresoras.

* La Unidad Central de Proceso o CPU

Es la parte mas importante. Tambien se conoce como microprocesador.La CPU esta formada a su vez por la unidad de control y la unidad aritmetica y logica.

La funcion de la CPU consiste en leer y escribir contenidos de las celdas de memoria, llevar y traer datos entre celdas de memoria y registros especiales y decodificar y ejecutar las instrucciones de un programa.

* Analizando las unidades de la CPU

Unidad de control: su funcion es intentar reconocer las instrucciones en binario y generar las ordenes para que se ejecuten.

Unidad aritmetico-logica: trata de realizar las operaciones basicas de calculo aritmetico,como puede ser la suma, resta, multiplicacion, o la division, asi como las operaciones logicas bit a bit, and, or , xor, not, neg. (lo aprenderas mas adelante)

Sobre decodificar y ejecutar las instrucciones de un programa:

El reconocimiento de las instrucciones (secuencias de valores binarios determinados) es muy importante, pues de no ser reconocida, en procesadores anteriores al 80286 provoca un cuelgue, o bien la ejecucion de operaciones inesperadas, algo que puede aprovecharse para la deteccion de un determinado procesador, o una determinada marca de procesadores.

Procesador: como trabaja

El procesador trabaja de manera distinta segun el modo de funcionamiento activado. Los modos existentes en los 80x86 son el modo REAL (para mantener la compatibilidad con los 8086), el modo PROTEGIDO (una extension muy amplia de este modo), y el modo intermediario entre los dos, el VIRTUAL 86 (V86). Dependiendo del modo de trabajo activo se realiza de una manera distinta el direccionamiento de la memoria, es decir que se interpretan de diferente modo las direcciones a memoria. Aunque existen mayores diferencias entre los modos, nosotros estudiaremos el mas sencillo, el modo REAL

El procesador cuenta con una serie de celdas de memoria que se utilizan con mucha frecuencia y que, como ya hemos visto, forman parte de la CPU.

Estas celdas son conocidas con el nombre de registros. Un procesador puede tener una docena o dos de estos registros. La unidad aritmetica y logica de la CPU realiza las operaciones relacionadas con los calculos numericos y simbolicos. Tipicamente estas unidades solo tienen capacidad de efectuar operaciones muy elementales como: suma y resta de dos numeros de punto fijo, multiplicacion y division de punto fijo, manipulacion de bits de los registros y comparacion del contenido de dos registros.

Un dato mas:

Las computadoras personales pueden clasificarse por lo que se conoce como tamaÏo de palabra, esto es, la cantidad de bits que el procesador puede

manejar a la vez.

0x03. Atacando al lenguaje

A.- Para empezar
Antes de seguir conviene hacer una aclaracion: cuando hacemos un programa, primero lo escribimos en un lenguaje de alto nivel (que entendemos) y despues lo traducimos a lenguaje maquina. Ya esta, podemos seguir.

La CPU tiene 14 registros internos, cada uno de 16 bits. Los primeros cuatro, AX, BX, CX, y DX son registros de uso general y tambien pueden ser utilizados como registros de 8 bits, para utilizarlos como tales es necesario referirse a ellos como por ejemplo: AH y AL, que son los bytes alto (high) y bajo (low) del registro AX. Esta nomenclatura es aplicable tambin a los registros BX, CX y DX. Esto se hace en procesadores x86

Los registros son conocidos por sus nombres espec¡ficos. Algunos de ellos:

AX Acumulador
BX Registro base
CX Registro contador
DX Registro de datos
DS Registro del segmento de datos
ES Registro del segmento extra
SS Registro del segmento de pila
CS Registro del segmento de c¢digo
BP Registro de apuntadores base
SI Registro ¡ndice fuente
DI Registro ¡ndice destino
SP Registro del apuntador de la pila
IP Registro de apuntador de siguiente instrucci¢n
F Registro de banderas

Mas adelante veremos mas concretamente los regsitros de banderas, pila,etc..

Algo practico: Desde MS-DOS, escribe debug: aparecera un guion (interprete de comandos) Si escribes "r", veras el registro.

B.- Registros "generales".
Estos registros son escribibles y legibles directamente mediante instrucciones de transferencia
de datos.

Registro	Ancho.	Nombre.
AX	16 bits	ACUMULADOR
BX	16 bits	
CX	16 bits	CONTADOR
DX	16 bits	
EAX	32 bits	(386+)
EBX	32 bits	(386+)
ECX	32 bits	(386+)
EDX	32 bits	(386+)

ALGO MAS COMPLEJO: a veces se quiere acceder a la parte baja y alta de uno deestos registros, por ello existen nombres para designar la parte baja rL, de 8 bits y la alta rH de 8 bits, donde r es A o B,o C o D. Por ejemplo, AL designa a la parte baja del registro AX.De igual manera existe AH,AL,BL,BH,CH, CL,DL y DH.

Toda direccion en modo REAL se descompone en un segmento y un offset, siendo la representacion grafica tipica SEG:OFFS. Dicha direccion es virtual, es decir que corresponde con una direccion fisica de longitud maxima 1Mb, y que calculamos mediante la formula siguiente:

16*SEG+OFFS = direccion fisica.

De esta manera al direccionar la direccion por ejemplo 0:100, estamos accediendo a la posicion de memoria 0*16+100, es decir a la 100. Se puede por lo tanto acceder a una misma posicion de memoria fisica(de 0 a 1Mb) de distintas maneras. Por ejemplo, acceder a la direccion 0:160h o 16h:0 implica acceder a la misma posicion fisica, la 160h. Si fijamos el segmento, el offset tiene el papel de indice dentro de un array de 64kb como maximo. Veamos varios ejemplos practicos, pero antes es necesario conocer lo que son los registros de segmento.

Dispone el procesador de unos registros de segmento, que almacenan el segmento que se va utilizar en una operacion de acceso a memoria, cargado por el intermediario de un registro general [no te desmoralices si no lo entiendes demasiado]

Si te fijaste, en el registro del debug habia unas letras al final. Esto es lo que quieren decir:

Overflow
NV = no hay desbordamiento;
OV = si lo hay
Que es el overflow? pues si un registro puede almacenar hasta un numero de 2 cifras, si introducimos uno de 3, se desborda.

Direction
UP = hacia adelante;
DN = hacia atras;

Interrupts
DI = desactivadas;
EI = activadas

Sign
PL = positivo;
NG = negativo

Zero
NZ = no es cero;
ZR = si lo es
Esto va a ser util a la hora de comparar numeros.

Auxiliary Carry
NA = no hay acarreo auxiliar;
AC = hay acarreo auxiliar

Parity
PO = paridad non;
PE = paridad par;

Sobre el bit de paridad: Como ya sabes, 1 byte = 8 bits (una letra). Pues para dar mayor seguridad, cada x cadenas(generalmente en todas) se a¤ade un noveno bit. Si al recibir todas las cadenas tenemosun numero par de bits de paridad, quiere decir que todo ha funcionado correctamente.

Carry
     NC = no hay acarreo;
     CY = Si lo hay

0x04. Haciendo el primer programa

   Si! Por fin! Pero.. (joooo!!!) Solo una cosa mas: en el lenguaje ensamblador se pueden apreciar dos partes en el codigo: la primera es el nombre de la instrucci¢n, y la segunda, los par metros de la misma. Por ejemplo: sabiendo que la instruccion "mov" mueve el parametro indicado al registro que mandemos, y que AX es el acumulador, se podria hacer lo siguiente:
En ms-dos, escribir "debug" teclear "a 0100 assamble".
Para ver una lista extendida de los comandos escribe "help" o bien "?"
Esta orden le indica al debug que queremos comenzar a indicar instrucciones en la direccion 0100 (os acordais de lo de las celdas?) Normalmente, al escribir "a" se colocara automaticamente en 0100, pero conviene recordarselo, ya que si no puede traer problemas.
Una vez hecho esto, teclear lo siguiente:

```
0CAE:0100 mov ax,0002
0CAE:0103 int 20
0CAE:0105
```

Comentarios: la instruccion mov ax, 002 mueve el valor HEXA 0002 a ax (primero le indicamos a donde lo movera, y despues que movera) Int 20 le indica al programa que hemos terminado.

Para ejecutar nuestro programa escribimos:

```
-g
```

   Nos saldr el siguiente mensaje:
"El programa ha terminado de forma normal"

  Si queremos comprobar el valor de ax tecleamos:

```
-g 01013
```

  (genera hasta 103)

Nos mostrara el registro:

```
AX=0002  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CAE  ES=0CAE  SS=0CAE  CS=0CAE  IP=0103   NV UP EI PL NZ NA PO NC
0CAE:0103 CD20          INT     20
```

   Los demas pueden cambiar, pero ax debe valer dos.

  Otras instrucciones:
La instruccion ADD, suma dos numeros.
Ejemplo:

```
ADD bx,ax  ;Esto sumara A BX el valor de ax (dos en este caso). Por lo
```

tanto, bx pasara a valer 2h*
*:la h se usa para indicar que es hexadecimal.

Ejercicios: Crea un progama que almacene en cx la suma de dos numeros cualesquiera,

```
estando x en ax, e y en bx (p.ej, ax=2,bx=3,cx=2+3)


SOLUCION:
  Un posible programa seria este:

-a
0CAE:0100 mov ax,0007   /indica que ax vale 7/
0CAE:0103 mov bx,0004   /indica bx vale 4/
0CAE:0106 add cx,ax     /queremos que ax y bx almacenen el valor de nuestros
numeros,
0CAE:0108 add cx,bx          /con lo  cual estos no se pueden usar para
almacenar la suma. Por
0CAE:010A int 20               /tanto le indicamos que a¤ada al valor de cx
(0000)) el valor de uno de ellos, para luego
0CAE:010C                    /sumarle el otro   /
-g

El programa ha terminado de forma normal

-g 010a

AX=0007  BX=0004  CX=000B  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CAE  ES=0CAE  SS=0CAE  CS=0CAE  IP=010A   NV UP EI PL NZ NA PO NC
0CAE:010A CD20          INT     20

Por que aparace CX=000B? Pues porque 000Bh(hexadecimal) = 0011d (decimal)
   Y ya est !




0x05. Guardando y cargando programas

A.- GUARDAR:
   Para guardar un programa se necesita saber el tama¤o del mismo, indicarle
a cx
lo que ocupa y guardar. Lo mejor es un ejemplo, con el programa anterior:


-a
0CAE:0100 mov ax,0007
0CAE:0103 mov bx,0004
0CAE:0106 add cx,ax
0CAE:0108 add cx,bx
0CAE:010A int 20
0CAE:010C
-n prueba.com
-h 10c 100
020C  000C
-rcx
CX 0000
:000c
-w
Escribiendo 0000C bytes

Explicamos:
n le indica el nombre. Estos archvos tienen la extension "com"
h nos indica el tama¤o que hay entre la celda 010c y 0100. El primer parametro
es la suma (100+ 10c) y el segundo la resta, es decir, el tama¤o de nuestro
programa.
rcx: nos muestra el valor del regisro cx y nos deja modificarlo. CX indica
```

el espacio que queremos reservar para nuestro programa, en este caso, 000c
w: (write) pues guarda. va a guardar en el primer sitio que pille, es decir, si en msdos antes de escribir debug aparece C:/ archivos de pepito, ahi se guardara. Si aparece solo c:/ pues en C...

B.- CARGAR:
  para cargar, primero le indicamos el nombre, y despues le decimos hasta donde cargamos.
Ejemplo:

```
-debug
-n prueba.com
-l
-u 100 10a

0CFA:0100 B80700        MOV     AX,0007
0CFA:0103 BB0400        MOV     BX,0004
0CFA:0106 01C1          ADD     CX,AX
0CFA:0108 01D9          ADD     CX,BX
0CFA:010A CD20          INT     20
```

la instruccion "l" (load) le indica que cargue el programa cuyo nombre introducimos.
"u" (unasamble) le dice que nos muestre lo que hay. para evitar ver lineas que no pertenecen a nuestro programa le indicamos que abra desde 100 hasta 10a. Y ya esta.

0x06. Condiciones, ciclos y bifurcaciones

A. Teoria
   Pues esto sirve si queremos hacer un programa que, por ejemplo, dependiendo de si quieres o no, te muestre un mensaje por pantalla. O si la contrase¤a es correcta o no te deja registrarte xD. He aqui cosas nuevas:

Instruccion	Funcion
jmp	salta a la direccion que le indiquemos e (direccion)
	almacena cadenas de texto. Ej: e 105 "hola" 0d 0a "$". 0d 0a es el valor hexa de intro (retorno de carro)
int 21	ejecuta una determinada funcion dependiendo del valor ah*. Por ejemplo,
	si ah vale 9, muestra un mensaje. (el que se encuentre en dx)
loop [valor]	loop trabaja con cx. Si cx es mayor que 0, salta a la direccion que le indiquemos
dec	resta uno a valor. Ej: dec cx
jnz [valor]	jnz trabaja con bx. Si bx es distinto a cero salta a valor
jcxz [valor]	si cx es cero, salta a valor. Se puede usar bx, ax....

*ah: este registro que nos aparece nuevo, indica lo que debe hacer int 21

B. Practica

La forma mas sencilla de comprender este tema es por medio de ejemplos. (UNI DE GUADALAJARA). Vamos a crear tres programas que hagan lo mismo: desplegar un n£mero determinado de veces una cadena de caracteres en la pantalla.

```
- a100
0C1B:0100 jmp 125 ; brinca a la direcci¢n 125H
0C1B:0102 [Enter]
- e 102 "Cadena a visualizar 15 veces" 0d 0a "$"
```

```
- a125
0C1B:0125 MOV CX,000F ; veces que se desplegara la cadena
0C1B:0128 MOV DX,0102 ; copia cadena al registro DX
0C1B:012B MOV AH,09 ; copia valor 09 al registro AH
0C1B:012D INT 21 ; despliega cadena
0C1B:012F LOOP 012D ; si CX>0 brinca a 012D
0C1B:0131 INT 20 ; termina el programa.
```

Por medio del comando "e" es posible introducir una cadena de caracteres en una determinada localidad de memoria, dada como par metro, la cadena se introduce entre comillas, le sigue un espacio, luego el valor hexadecimal del retorno de carro, un espacio, el valor de linea nueva y por ultimo el simbolo ¡$¡ que el ensamblador interpreta como final de la cadena. La interrupcion 21 utiliza el valor almacenado en el registro AH para ejecutar una determinada funcion, en este caso mostrar la cadena en pantalla, la cadena que muestra es la que est almacenada en el registro DX. La instruccion LOOP decrementa automaticamente el registro CX en uno y si no ha llegado el valor de este registro a cero brinca a la casilla indicada como parametro, lo cual crea un ciclo que se repite el numero de veces especificado por el valor de CX. La interrupcion 20 termina la ejecucion del programa.

Otra forma de realizar la misma funci¢n pero sin utilizar el comando LOOP es la siguiente:

```
- a100
0C1B:0100 jmp 125 ; brinca a la direccion 125H
0C1B:0102 [Enter]
- e 102 "Cadena a visualizar 15 veces" 0d 0a "$"
- a125
0C1B:0125 MOV BX,000F ; veces que se desplegara la cadena
0C1B:0128 MOV DX,0102 ; copia cadena al registro DX
0C1B:012B MOV AH,09 ; copia valor 09 al registro AH
0C1B:012D INT 21 ; despliega cadena
0C1B:012F DEC BX ; decrementa en 1 a BX
0C1B:0130 JNZ 012D ; si BX es diferente a 0 brinca a 012D
0C1B:0132 INT 20 ; termina el programa.
```

En este caso se utiliza el registro BX como contador para el programa, y por medio de la instrucci¢n "DEC" se disminuye su valor en 1. La instruccion "JNZ" verifica si el valor de B es diferente a 0, esto con base en la bandera NZ, en caso afirmativo brinca hacia la direcci¢n 012D. En caso contrario contin£a la ejecuci¢n normal del programa y por lo tanto se termina.

Una £tima variante del programa es utilizando de nuevo a CX como contador, pero en lugar de utilizar LOOP utilizaremos decrementos a CX y comparacion de CX a 0.

```
- a100
0C1B:0100 jmp 125 ; brinca a la direcci¢n 125H
0C1B:0102 [Enter]
- e 102 "Cadena a visualizar 15 veces" 0d 0a "$"
- a125
0C1B:0125 MOV DX,0102 ; copia cadena al registro DX
0C1B:0128 MOV CX,000F ; veces que se desplegara la cadena
0C1B:012B MOV AH,09 ; copia valor 09 al registro AH
0C1B:012D INT 21 ; despliega cadena
0C1B:012F DEC CX ; decrementa en 1 a CX
0C1B:0130 JCXZ 0134 ; si CX es igual a 0 brinca a 0134
0C1B:0132 JMP 012D ; brinca a la direcci&oaute;n 012D
0C1B:0134 INT 20 ; termina el programa
```

En este ejemplo se usa la instruccion JCXZ para controlar la condici¢n de salto,

el significado de tal funcion es: brinca si CX=0

El tipo de control a utilizar depender de las necesidades de programaci¢n
en determinado momento.

Ejercicios: En funcion de lo visto deberias ser capaz de crear un programa
que imprima tu nombre UNA vez por pantalla. Te crees capaz?

Solucion:
Una posible solucion seria esta:

```
-a 100
0CAE:0100 jmp 125
0CAE:0102
-e 102 "Hola, soy x" 0d 0a "$"
-a 125
0CAE:0125 mov dx,0102
0CAE:0128 mov cx,0001
0CAE:012B mov ah,09
0CAE:012D int 21
0CAE:012F int 20
0CAE:0131
-g
Hola, soy x

El programa ha terminado de forma normal
```

0x07. Alguna cosilla mas

   A la hora de crackear (Si ese es el objetivo!) debes saber lo que es la pila
(donde se almacenan las cadenas) y la instruccion "cmp" esta compara dos valores
y en funcion de esto hara una cosa o no (te dejara registrarte...)

  Si te interesa este tema, te aconsejo de que busques guias que te ayuden mas
en el mundo del assambler, esto es solo el principio....

  Se me olvidaban los saltos condicionales! Pues estos hacen x cosa en funcion
de como haya ido la comparacion. Ejemplo:

```
CMP EAX,EBX
JE 0102
```

  Mande?? Si, veras: compara EAX con EBX. Estos podrian contener por ejemplo
el serial bueno y el que nosotros hemos introducido. Si son iguales (Jump if
Equal) pues salta a 102, donde podria comenzar el c¢digo que nos dice que
estamos registrados. Esto es solo un ejemplo.

Aunque podr¡a saltar siempre (JMP) Saltar si vale cero (JZ) si no vale cero
(JNZ) si no son iguales... (JNE) etc etc etc. Pero esos son los mas usados.

0x08. Cracking.

A.Tecnicas de proteccion.

  Pues hay pogramas que hay que pagar para usar. Te dejan un tiempo para que
los pruebes, y despues... se acabo. Por supuesto, no se borran, ya que ES ILEGAL
que un programa borre archivos sin la autorizacion del usuario. He aqui los

sistemas de proteccion mas comunes (por d4rkm4st3r, numero anterior):

****
********** Sistemas de Proteccion Por Tiempo**********
****

Los sistemas de proteccion por tiempo pueden operar de distintas formas:

*El software COMPRUEBA SI HAN TRANSCURRIDO X CANTIDAD DE DIAS desde la instalacion del mismo, y si es asi el software procede a su salida inmediata. Durante la salida del software este puede mostrar algun mensaje informando al usuario del hecho en cuestion (The Evualuation Period has expired).

*El software COMPRUEBA SI HA LLEGADO A UNA FECHA LIMITE, si es asi procede de la misma manera que en el caso anterior. La diferencia esta en que el software dejara a partir de una fecha determinada y no volvera a funcionar si se vuelve a instalar.

****
********** Sistema de Proteccion Por Nags**********
****

Este sistema no es propiamente un sistema de proteccion, sino es un sistema para molestar al usuario del software y recordarle q debe adquirir el software original. Estas son cuadros de dialogo q aparecen al inicio o al final de la aplicacion y estan activos hasta q el usuario un determinado boton o se termina una cuenta atras.

****
********** Sistema de Proteccion Por CD-Check**********
****

Este sistema de proteccion comprueba que el CD del software se encuentra en la unidad de CD-ROM cada vez q se ejecuta. Estos sistemas no evitan q el CD sea duplicado, lo q significa q si introducimos una copia funcionara normalmente. Hay dos variantes de este sistema: en el primero identifica al CD-ROM mediante la etiqueta q este posee, si el CD tiene la etiqueta esperada continua su ejecucion. El segundo metodo se ocupa de comprobar si existe un determinado archivo dentro del CD-ROM, si el archivo existe, continua con la ejecucion. La dificultad de estos sistemas va aumentando cuando si necesita los datos que hay en el CD para poder continuar.
  ***Los juegos, lo usan para que necesariamente tengas que introducir el CD, es decir, que no baste con que te instales el juego de un amigo....

B. Atacar
   Pues el programa no se va a dejar atacar facilmente. Estas son las tecnicas que usara para protegerse:
*Anti-Debugging: El software usa tecnicas para detectar nuestra presencia y evitar ser trazado (significa correr un fragmento de la aplicacion con un debugger). Normalmente Intentara detectar la presencia de un debbuger y parar la ejecucion del software mientras este presente.

*Encriptacion/Compresion de datos: El software utiliza tecnicas q ocultan el verdadero codigo del programa hasta q se ejecuta, inutilizando asi cualquier intento de desamblado del codigo.

  Para atacar un programa lo mejor es usar las cadenas conocidas (pincipalmente mensajes de texto). Lo veras en las practicas.

C. Practicar:
   Vas a necesitar un debug (el olly, p.ej) y un editor de textos hexadecimal. para que el editor? para cambiar los valores, esto no lo hace el debug. Para que el debug? para ver el programa!!

Esta lecci¢n corre de tu cuenta. Pero, crackmes sencillios hay...
Los crackmes son programas para que la gente practique, y amplie sus conocimientos. Sin ir m s lejos, en el n£mero anerior de SET te explican uno paso a paso.

C bis. Practicar
   Esto cuenta el doble. Ademas, no te quedes en este texto continua en los siguientes....

0x09 Despedida y Reflexion

   Si este texto llega a publicarse, se confirmara la teoria de que la "scene" esta en crisis. Hay gente muy buena ahi fuera que posee grandes conocimientos como para que se tengan que publicar cosas con un nivel... bajo (en relacion al habitual) Cualquiera que se haya dado una vuelta por la web habra podido comprobar de que hablo.

   Con esto simplemente espero que nos demos cuenta (yo tambien) de que la scene no son solo los mas "31337", somos todos. Y en SET no solo escriben 3, todos los lectores pueden hacerlo (mirarme a mi...) Todos los que se quejan del nivel: ¨Por que no escribis? Comprendo que esteis hasta arriba de examenes (incluso... 4?6 meses seguidos?) Pero... y los que escriben?
¨No tienen obligaciones? ¨Viven de esto? ...
Yo, desde luego, no estoy dispuesto a dejar que una de las mejores publicaciones que ha existido/existe se derrumbe por vagueza. Y si hace falta escribir cosas sencillas, que reunan otros texos, se escriben. Ya iremos aprendiendo. Pero ser solidarios, y participar (parece una campa¤a del Domund xD)

   Nota final: Como dije al principio es una recopilacion de textos que circulan libremente por internet. -Que os haya servido de ayuda!

¨SIGNIFICA ESTO QUE LA AUDIENCIA PIDE UN NIVEL MAS SENCILLO? (la audiencia es la que "manda", y si no escribe, es porque no entiende?)

"Alter, Glaube, Geld, Rasse... Was macht das?
(Edad, pasta, raza, religion.... Todo eso que importa?)

En fin, el caso es que espero que os haya servido
Un saludo de ThEnemI.

EOF

```
-[ 0x0E ]-----------------------------------------------------------------
-[ Un poco de historia ]--------------------------------------------------
-[ by blackngel ]---------------------------------------------------SET-29--
```

blackngel_hack@hotmail.com

HISTORIA
"Bienvenidos al mundo de lo real..."

```
@#@#@#@#@================@#@#@#@#@
@#@#@#@#@    0.INDICE    @#@#@#@#@
@#@#@#@#@================@#@#@#@#@
```



```
@#@#@#@#@================@#@#@#@#@
@#@#@#@#@ 1.INTRODUCCION @#@#@#@#@
@#@#@#@#@================@#@#@#@#@
```

Hola de nuevo a tod@s, ahora estoy aqui no para dar explicaciones de seguridad o nuevos bugs que hayan salido (no sera porque sean pocos :)).

Hoy voy hacer un recordatorio desde las primeras epocas de los sistemas telefonicos e informaticos hasta la fecha de hoy sobre los actos mas importantes que han pasado acerca del mundo de la informatica y el underground y hare unas peque¤as menciones a ciertos grupos de personas (hackers) que han quedado enmarcados en nuestra memoria por sus diferentes actividades en este mundillo.

Creo que no hara falta mas introduccion para lo que me propongo, espero que os haga recordar algo y si podeis o sabeis de algun detalle importante en alguna fecha pues me lo hagais saber atraves de mi direccion de correo que como ya sabeis es: blackngel_hack@hotmail.com y tendre muchisimo gusto en a¤adirlo y exponer la actualizacion otra vez.

```
@#@#@#@#@================@#@#@#@#@
@#@#@#@#@   2.HISTORIA   @#@#@#@#@
@#@#@#@#@================@#@#@#@#@
```

1878 -->> - Menos de dos a¤os despues de que el sistema telefonico de

Alexander Graham Bell empezara a funcionar, un grupo de adolescentes estropeo la red.

1958 -->> - EE.UU. crea ARPA (Advanced Research Projects Agency), ciencia y tecnologia aplicada al campo militar.

1960 -->> - Los hackers originales utilizaron los primeros mainframes del MIT para desarrollar habilidades y explorar el potencial de la informatica. En esa epoca, "hacker" era un termino elogioso para los usuarios con un conocimiento exponencial de los ordenadores.

1969 -->> - La agencia de proyectos de investigacion avanzados del Departamento de Defensa (DoD), construyo Arpanet.

1971 -->> - Antes del uso masivo de los ordenadores y de Internet, los "phreakers" utilizaron la extensa base de redes telefonicas. John Draper, AKA Cap'n Crunch, descubrio que un simple silbato permitia a los usuarios entrar en los sistemas de facturacion de las llamadas a larga distancia.

1973 -->> - Kahn desarrolla un nuevo protocolo, el TCP/IP (Transmisi¢n Control Protocol/ Internet Protocol).

1976 -->> - Dos miembros del Homebrew Computer Club lanzaron las llamadas "blue box", que se utilizaban para hackear sistemas telefonicos. La pareja (Steve Jobs y Steve Wozniak) conseguirian hacerse famosos despues al fundar Apple Computer.

1983 -->> - Primer arresto de hackers por el FBI despues de que invadieran el centro de investigacion de Los Alamos.

- Se estrena la pelicula "Juegos de guerra", que cambio la percepcion del publico con relacion a los hackers y establecio su prestigio.

1984 -->> - Se funda la publicacion trimestral 2600 (nombrada como la frecuencia del silbato de John Draper), que ofrecia una plataforma a los hackers y phreakers.

- Se forma Legion of Doom (LoD)

1987 -->> - Herbert Zinn, de 17 a¤os de edad, es arrestado despues de entrar en el sistema de AT&T. Los expertos afirman que estuvo a punto de bloquear todo el sistema telefonico norteamericano.

- Se crea el primer virus conocido de MS-DoS, "Brain". Los investigadores creen que se escribio en Pakistan. Infectaba el sector de arranque de los disquetes de 360 KB.

1988 -->> - Robert Morris bloquea 6.000 ordenadores a traves de ARPANET con su famoso virus, que lanzo, segun, propias palabras, de forma accidental.

- Se funda la CERT (Computer Emergency Response Team).

- Aparece el primer software antivirus, escrito por un desarrollador de Indonesia.

1989 -->> - Primer caso de ciberespionaje en Alemania Occidental.

- The Mentor lanza el manifiesto "Conscience of a Hacker", que

finaliza con una frase inquietante: "Pueden detener a una persona, pero no pueden detenernos a todos".

1990 -->> - Se lanza el grupo de apoyo "Freedom on the Internet".

- Aparecen sofisticados tipos de virus como los poliformicos (que se modifican a si mismos cuando se expanden) o los de multiparticion (que infectan diversas zonas de una maquina).

- El First National Citibank de Chicago sufre el primer robo informatico reconocido por una cantidad de 70 millones de dolares.

- El hacker Dark Dante, Kevin Lee Poulsen, es arrestado despues de una busqueda de 17 meses. Robaba secretos militares.

- Mitnick y Shimomura miden sus fuerzas.

1993 -->> - Se celebra la primera conferencia Def Con de hacking en Las Vegas. Se suponia que el evento era una celebracion unica para decir adios alas BBS (obsoletas por la Web), pero resulto tal exito que se convirio en evento anual.

1994 -->> - Hackers atacan a los sitios web federales de los EE.UU., incluyendo la CIA, el Departamento de Justicia, la NASA y la Fuerza Aerea. No fue la mejor forma de hacerse popular entre los estamentos militares.

- Vladimir Levin, el legendario lider de un grupo de hackers ruso, parece ser el cerebro del robo virtual de 10 millones de dolares de Citibank. Fue arrestado en Londres un a¤o despues y extraditado a EE.UU.

1995 -->> - El Departamento de Defensa de EE.UU. sufre 250.000 ataques en un a¤o.

- Mitnick es arrestado bajo sospecha de robar 20.000 numeros de tarjetas de credito. Es encontrado culpable un a¤o despues.

- La pelicula "Hackers" llega a las pantallas de cine, difundiendo algunas ideas equivocadas sobre las actividades de los hackers.

1998 -->> - Network Associates emite un anuncio anti-hacker durante la Superbowl en los EE.UU. En el, dos tecnicos de misiles sovieticos destruyen el mundo, inseguros de saber si las ordenes vienen de Moscu o de los hackers.

- Los hackers afirman haber entrado en el sistema de satelites militares y amenazan con vender secretos a los terroristas.

- Se crea la NIPC (National Infrastructure Protection Centre) con un presupuesto multimillonario.

- El grupo L0pht declara que puede paralizar Internet en media hora.

1999 -->> - Nacimiento del software anti-hacking.

2000 -->> - Ataques de denegacion de servicio (DoS) sobre los grandes nombres de la Red.

2001 -->> - XP, "el Windows mas seguro", es crackeado antes de su lanzamiento.

2002 -->> - Bill Gates, el jefe de Microsoft crea Trustworthy Computing.

2002 -->> - El ISP CloudeNine es "hackeado hasta la muerte".

@#@#@#@#@================@#@#@#@#@
@#@#@#@#@   3.GRUPOS   @#@#@#@#@
@#@#@#@#@================@#@#@#@#@

////////// LEGION OF DOOM (LoD) //////////

Este grupo de culto de los hackers se convirtio en un guardian con el lanzamiento de la empresa de seguridad ComSec. Algunos de los componentes que no entraron en el nuevo mundo de ComSec terminaron en prision despues de una prolongada guerra con los Masters of Deception.

////////// COMPUTER UNDERGROUND //////////

Un grupo casi esoterico dedicado a promocionar el libre intercambio de informacion, sobre todo con relacion a la alta tecnologia.

//////////   L0PHT   //////////

Fundado por un grupo de amigos en un loft de Boston -de ahi el nombre- L0pht alcanzo la prominencia cuando advirtieron a Washington que podrian paralizar Internet en media hora a menos que se mejoraran las medidas de seguridad del Gobierno. El grupo afirma que solo hackea para detectar los "agujeros" que pueden presentar las empresas o los departamentos gubernamentales. Ha lanzado herramientas anti-hacking como AntiSniff, que controla las redes para evitar que se utilicen en ellas herramientas sniffer.

////////// MASTERS OF DECEPTION //////////

Famoso por haber violado la seguridad de Bank of America, AT&T y la NSA, los Masters of Deception vieron la luz despues que Mark Abene (Phiber Optik) fuese expulsado de Legion of Doom. Los dos grupos se enzarzaron en una ciberguerra por un comentario racista sobre John Lee (MoD) y muchos creian que la batalla iba a remitir cuando la mayoria de los miembros de estos grupos fueron arrestados en 1993.

MoD era un grupo dedicado al profundo conocimiento del sistema telefonico.

////////// CULT OF THE DEAD COW //////////

Este grupo de hacking lanzo el programa "Back Orifice" (Un troyano) una potente herramienta de hacking en Def Con. Una vez que se instalaba el programa en una maquina Windows 95/98, el hacker podia tener acceso remoto no autorizado al sistema.

//////////   YIHAT   //////////

Kim Schmitz lidero un grupo de hackers denominado YIHAT (Young Intelligent Hackers Against Terrorism), que afirmo haber tenido acceso a las cuentas bancarias de Osama Bin Laden en los dias posteriores al 11 de Septiembre

del 2001. La afirmacion se demostro falsa y el lider del grupo fue
extraditado desde Tailandia a Alemania por una serie de delitos financieros.

////////// CHAOS COMPUTER CLUB //////////

El grupo aleman Chaos llego a los titulares con una demostracion televisiva
en directo del robo de una cuenta bancaria, utilizando un simple control
Microsoft ActiveX. El Club configuro un control ActiveX para que ejecutase
una transferencia fraudulenta a un PC que ejecutaba una aplicacion financiera.
Microsoft se¤alo que muchos ejecutables podrian conseguir el mismo resultado.

@#@#@#@#@================@#@#@#@#@
@#@#@#@#@ 4.PERSONAJES @#@#@#@#@
@#@#@#@#@================@#@#@#@#@

Esto es un peque¤o recordatorio de ciertas personas que han quedado en la
mente de muchos de nosotros y que han contribuido fielmente al desarrollo
del mundo informatico, ellos nos han revelado sus entra¤as.

\\\\\\\\\\   RICHARD STALLMAN   \\\\\\\\\\

Fue un estudiante de Harvard, consiguio un puesto en el laboratorio de
inteligencia artificial del MIT. Odiaba la idea del software de propiedad
privada, por este motivo, mas adelante creo la FSF (Free Software
Foundation).

Despues de retirarse de los laboratorios del MIT, alla por los 80,
desarrollo un Sistema Operativo llamado GNU (Gnu is not Unix).
Tambien se encarga de la creacion de muchas utilidades gratuitas para
entornos UNIX.

No estaria de mas que os dieseis una vuelta por la pagina de la FSF.

\\\\\\\\\\   KEVIN MITNICK   \\\\\\\\\\

Simplemente el mas buscado por todos los cuerpos del FBI, sus acciones
han sido ejemplo de mucha de gente, algunos lo consideraron el chico
perdido del ciberspacio.

Con 10 a¤os burlo el sistema de seguridad de la defensa de los Estados
Unidos. Cazado a la vez por otro hacker llamado Tsutomu Shimomura
despues de que un feliz dia de navidad mitnick penetrara sus ordenadores
en busca de un software de OKI que hacia que su telefono fuera invisible
para los cuerpos del FBI.

Los primeros ordenadores que fueron acariciados por las preciadas manos
de Mitnick fueron los de las tiendas de "Radio Shack".
En la actualidad no puede tocar cualquier ordenador o telefono.

Conocido como el "Condor", Kevin realizo un tratamiento para intentar
superar su adiccion a los ordenadores. Hasta donde podiamos llegar...
Entonces creo que a la mitad de nosotros (me incluyo) nos tendrian que
hacer lo mismo. Soy un adicto!!!

Kevin Mitnick, hacker o cracker?, se admiten opiniones...

\\\\\\\\\\ ROBERT MORRIS \\\\\\\\\\

En el a¤o 1988 puso en la red a un gusano que infecto 6000 ordenadores aprovechandose de una vulneravilidad de nuestro amigo "SendMail", y atraves de la red ARPANET.

Toco su primer ordenador cuando su padre, jefe del NCSC, le trajo una maquina directamente desde la NSA.

Morris realizaba muchas de sus acciones desde una cuenta que poseia en los sistemas Bell y en los cuales consiguio acceso de Super Usuario.

\\\\\\\\\\ KEVIN POULSEN \\\\\\\\\\

Mas conocido como Dark Dante se hizo famoso por el control que hizo de las llamadas que se realizaban a la emisora de radio KIIS-FM, asi hizo que su llamada fuera exactamente la numero "102" consiguiendo de esta forma un Porsche 944 S2.

Era poseedor de un TRS-80 y fue declarado culpable por la busqueda de informacion secreta sobre los angentes del FBI.

\\\\\\\\\\ VLADIMIR LEVIN \\\\\\\\\\

Salto a la fama por su gran logro al robar 10 millones de dolares al prestigioso "Citibank", penetro usando el ordenador de su despacho en la universidad.

Arrestado en Londres. Segun Levin, uno de sus abogados era un agente del FBI.

Una cabeza matematica.

\\\\\\\\\\ MARK ABENE \\\\\\\\\\

Su apodo, Phiber Optik, fundador y miembro de Masters of Deception. Inspiro a miles de personas en los Estados Unidos a explotar los sistemas de la red telefonica.

La revista New York califico a Mark como uno de los 100 hombres mas inteligentes de la ciudad.

Tubo en sus manos bastantes maquinas, siendo solo de su propiedad una TRS-80, las otras fueron:
- Apple II
- Timex Sinclair
- Commodore 64

\\\\\\\\\\ DENNIS RITCHIE \\\\\\\\\\

Este excelente programador ha sido el creador del lenguaje C, uno de los mas potentes que existen en el mundo de la programacion. Trabajador de Bell Labs.

En la actualidad trabaja en el desarrollo de un nuevo Sistema Operativo llamado "Plan 9" que es una evolucion superior del sistema Unix, pero con una extructura totalmente diferente ya que se ha tenido que reescribir desde el principio.

\\\\\\\\\\ JOHN DRAPER \\\\\\\\\\

Su nick, "Capitan Crunch", su aza¤a mas destacable fue la de utilizar un silbato de los cereales Crunch (de ahi su nombre), para generar un sonido de 2.600 Hertzios el cual causaba el corte de los contadores de la red telefonica de Bell.

John creo la primera "Blue Box", esta generaba el mismo tono y fue distribuida a muchisima gente que se aprovecho de su utilidad.

@#@#@#@#@==========@#@#@#@#@
@#@#@#@#@ 5.DESPEDIDA @#@#@#@#@
@#@#@#@#@==========@#@#@#@#@

Vaya vaya, si hasta da la impresion de que el escribir no cansa nada, pues no es asi y por este motivo me voy llendo que me duelen los dedos y necesito descansar la mente.

Espero que os haya gustado y que mas de alguno soltara una lagrimilla ;)

Sugerencias, dudas, a¤adidos para el articulo e insultos que merezcan la pen ser publicados, mail-me a: blackngel_hack@hotmail.com

by blackngel

EOF

```
-[ 0x0F ]--------------------------------------------------------------
-[ La Revolucion de los ordenadores Cuanticos ]------------------------
-[ by Hacke_ReNg0 ]---------------------------------------------SET-29--
```

La Revolucion de los ordenadores Cuanticos

por

[]Hacke_ReNg0[] para SET

"El orgullo no es una virtud, pero es padre de muchas de ellas"
...Asi que, señores, nunca renuncien a su orgullo, es la base de su motivacion y de lo que les da ganas de hacer cosas.
Y estoy muy orgulloso de haber escrito este articulo para que los que no saben aprendan!
Espero que otros se animen a escribir, asi hacen correr la informacion.

Gracias. []Hacke_R3Ng0[]

INDICE
,,,

,,,

1. Introduccion al articulo
 ~~~~~~~~~~~~~~~~~~~~~~~~

Bueno, me presento, es la primera vez que me largo a hacer un articulo para una revista asique tengan piedad.
Voy a decir a lo que voy con este articulo y porque me lanze a escribirlo.
Creo queo llege a un punto en el que pense que ya habia asimilado conocimientos necesarios para, ademas de aprender, enseñar a otros. Pense en sobre que podia escribir, y bueno, me acorde que unos meses atras habia leido un informe sobre la cuantica. Y me pregunte porque nadie publicaba articulos sobre este tema, entonces lo elegi.

Este articulo apunta a cual es la importancia de la cuantica en el futuro, (cuando se logre crear un computador asi) como se desarrolla, y cuales son
 ~~~~~~~~~~~~~~~~~~~~~~~~

las dificultades que, por ahora, impiden crear una computadora cuantica y que hay que solucionar. Espero que el articulo sirva para prevenir a algunos que no conocen esto, porque la cuantica es el futuro de las microcomputadoras, esas que las vas a tener en la mano y que las veran nuestros hijos o quizas nosotros, si tenes entre 15 años como yo.

2. Introduccion a la cuantica
   ~~~~~~~~~~~~~~~~~~~~~~~~~

a.Que es la programacion cuantica o los ordenadores cuanticos?
  ~~~~~~

Bueno, cabe aclarar que yo voy a explicar que es y que se podra hacer con una computadora PROGRAMADA cuanticamente, no la programacion cuantica en si. Lo que si voy a explicar es como funciona este sistema.
La cuantica es un mecanismo que se debera implementar en el futuro, si quieres tener una computadora del tamaño de la palma de tu mano. Es debido a que los microchips se estan haciendo cada vez mas pequeños y se necesita un metodo para hacerles soporte. Desde la creacion de los computadores que funcionaban con valvulas, hasta ahora, todos los componentes se fueron achicando, y lo seguiran haciendo, pero tienen una frontera que no puede ser traspasada: El mundo Subatomico. PERO LA CUANTICA SI PUEDE TRANSPASAR ESTA FRONTERA
Tal vez pienses que me fui por las ramas, pero es el cimiento que les debo hacer para que puedan entender que es la cuantica, debido a que es dificil explicar un termino que poco corre por internet.

Considero que hasta aqui ha quedado claro a que se refiere la cuantica. Mas adelante explicare detalladamente su formacion.

b. A que apuntan?
   ~~~~~~~~~~~~~~~

Como dije antes, los componentes electronicos se achicaran cada vez mas hasta que se topen con las particulas subatomicas. Paso a explicar con un grafico:

```
        [COMPONENTE]              |
        [MOLECULAS QUE LO FORMAN] | Estado normal de las cosas. Hasta aqui se
        [ATOMOS]                  | podran achicar los microcomponentes.
        ..........................
        [SUBATOMICOS]             | Aqui entra en juego la cuantica
```

Una de las ventajas de un futuro computador o sistema cuantico puede ser que las notebooks podran ser aun mas chicas que las de ahora, y mas potentes que un Athlon o una Pentium IV o V. Sera una nueva era de las computadoras, como el salto de las valvulas al sistema binario. Y para que empiece se calculan diez años.

Una computadora cuantica podra desencriptar el criptograma DES (utilizado para las transmisiones bancarias) en solo 5 minutos. No solo ese criptograma, sino cualquier otro en cuestiones de segundos. ¨Estan pensando lo mismo que yo?!xD O sea que la seguridad sufrira, pero tambien se perfeccionara con este nuevo sistema. Todo el software y el hardware sera mas seguro y funcionara mucho mejor. Por ej, un cortafuegos quizas pueda realizar miles de operaciones de seguridad a la vez. Segun la cuantica un software no tendra muchos agujeros en sus programas. Hablare de esto mas adelante. Tambien cabe decir que es el rumbo que hay que tomar para que en un futuro exista la IA, de la que tanto se habla.

c. Su importancia en la evolucion de las computadoras
   ~~~~~~

No me explayare mucho en esto, sin duda las computadoras seran mil veces mas rapidas y por razones que las veremos mas adelante. Tampoco tendran fronteras, se podra hacer lo que sea, con la potencia que sea y del tama~o que sea. Eso si un microondas ya no sera un aparato electrodomestico, sera una computadora, en esencia.

3. Investigacion cientifica
   ~~~~~~~~~~~~~~~~~~~~~~~~
a. Repasando el sistema binario
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Si llegaste hasta aca, supongo que sebras a lo que me refiero. Vamos a repasar

El sistema binario tiene dos estados: 0 y 1 (Encendido o apagado).
Este sistema opera por medio de los "operadores logicos", que comparan y resuelven por ej: 001001011 OR 011001010 = 000101100101. El operador OR toma dos estados, los compara y devuelve un resultado. Si alguno de los dos devuelve verdadero(1), el resultado sera verdadero(1), si los dos devuelven negativo(0), sera negativo. No tengo ganas de convertir este ejemplo a decimal, pero supone que de asi: 20 OR 43 = 102

Este sistema no hace operaciones como suma, resta, mult, division..., sino que hace solamente operaciones logicas. Esto lo hace la programacion, y aqui ya entramos en lo que es el software.
Ahh, y algo que casi se me pasa. Repasemos que es 32 bits o 16 bits. Una computadora de 32 bits quiere decir que el procesador transfiere secuencias de 32 bits a la velocidad reloj del mismo (233 Mhz, pentium, la mejor maquina de la historia!!, mi maquina que me esta durando 7 añoss!!).
Aqui termino con el sistema binario.

b. Que es y como funciona detalladamente?
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
En el sistema cuantico, los qubits (los veremos ahora) se mantienen en 2 estados simultaneamente (0 y 1), que sube notablemente el su velocidad de procesamiento. En cambio, en el sistema binario los datos son transmitidos si o si en 2 estados 0 o 1, no pueden estar en dos estados a la vez, lo que reduce la velocidad.

De BITS a QBITS:

Como vimos antes las computadoras de ahora estan basadas en la transferencia de bits logicos. Bueno, las computadoras cuanticas interactuan con las particula subatomicas que se basan en QUBITS. Este es el elemento basico de la computacion cuantica. Un QUBIT (Quantum bit o bit cuantico) presenta dos estados a la vez, 0 y 1 logico. Esto le da muchisima velocidad a la resolucion de algoritmos que en computadoras normales puede tadar miles de años y que en una cuantica solo unos segundos.

O sea, dos qubits tienen los estados: 00, 01, 10 y 11. 3 Qubits representan: 000, 001, 010, 100, 111, 101 y 110, y asi sucesivamente. El QUBIT no puede ser copiado, clonado o movido. Entiendan lo que serian 300 QUIBITs. La cantidad de estados que podrian tener a la vez teniendo una velocidad de transmision reloj de solo 200 MHz. Ya sobrepasaria la velocidad de una Athlon. Otra cosa importante es que el Qubit se basa en orientaciones magneticas de un atomo, no como el bit que se maneja con pulsos electricos. A los diversos estados que pueden tomar los Qubits se les llama "Superposicion".

Puede que se hallan quedado pensando en que los Qubits no pueden ser clonados, copiados o enviados. Y diran "y entonces de que sirven??". Pero hay un metodo que se llama "Tele transportacion" por la cual se pueden transmitir qubits sin enviarlos. Lo que se hace es esto:
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
COMENTARIO: Dale una leida rapida, despues otra para entender mas, y despues
otra tercera para analizar cada parte y que lo encuentres coherente

R: Qubit receptor | E: Qubit emisor | M: Qubit a transmitir
No importa si no entendes, segui leyendo.
============================================================================
'Los Qubits R y E son ENTRELAZADOS, MEZCLADOS o ENREDADOS y despues SEPARADOS

                       &R&E& ===> R E

'Luego el E es puesto en el emisor con el M y el R en el receptor

Emisor:ME                                                  Receptor: R

' M y E son leidos y cambian de estado debido a las interacciones anteriores.
' El estado es aleatorio.
' ~ = "informacion leida", ZZ= "Enviar"
' La informacionl eida es enviada al receptor.
COMENTARIO: Fijense que es enviada la INFORMACION, NO el QUBIT en si

Emisor: ~ME ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ Receptor: R~

' Al Qubit R se le aplica esa informacion y queda siendo una replica exacta de
' de Q

                        Receptor: R~= Q
=============================================================================


Veamos otro ejemplo para dejar bien claro lo de los estados del Qubit

Datos   | Estados posibles
2 bits  | 0 y 1
2 Qbits | 01, 10, 11, 00
```

Como ejercicio, para ver si entendiste quiero que te tomes la molestia de mandarme por email los estados posibles de una secuencia de 4 Qubits, tomando de referencia el cuadro y lo que aprendiste.

Considero que se sabe ya como funciona el sistema cuantico.

c. Las dificultades que se presentan para la construccion de un sistema cuantico

Los problemas son pocos pero algo dificil de resolver.
Un procesador, falla continuamente sin que nos demos cuenta. El sistema cuantico no acepta errores por los operadores logicos que son usados. Algo que lei en algun articulo dice: "En el sistema cuantico la Observacion MODIFICA el Resultado". Al ser OBSERVADO algo, si esto da error no da el Resultado.

Entonces el mayor problema es la falta de algoritmos inteligentes de operadores logicos con los cuales manejar la cuantica. Son demasiado viejos y primitivos para manejar algo tan complejo como la cuantica donde los datos se manejan parelelamente, las funciones se hacen todas a la vez, y asi los datos estan en muchos estados a la vez.

O sea que por ahora el hombre esta buscando algoritmos inteligentes para su manejo. Se espera que cuando se logre, se piense en un ordenador cuantico que opere sobre un SO. Mas vale apurarse a compatibilizar un SO que no sea Win con la cuantica (porque como dije antes la cuantica no acepta desestabilidades), y que ese ordenador, pueda desencriptar el Criptograma mas complejo que exista.
En realidad vamos a estar en lo mismo que ahora porque paralelamente se haran

criptogramas basados en cuantica que estaran a la altura de ella. Ej:

Xasdsdkoldñzxclmafcadfvanf~12zs ======> Criptograma de mi cuenta bancaria (DES)-
Viene un ordenador cuantico y me lo desencripta en 5 segundos: 45264796

Pero para solucionarlo, viene otro ordenador cuantico y lo encripta con su sistema:

XaSOzxAOakAAGvBBX2ZSZxx"x2$ai990123'10121"R$$"%$/(""#$#%$

Ahora viene el otro ordenador cuantico, agrandado, a desencriptarlo y se encuentra con que es mucho mas dificil de desencriptar.
Despues de dos años por fin:

45264796

Entienden?

NOTA: Las claves encriptadas no son nada que ver con el DES ni nada, es para ejemplificar.

Tambien, otro problema es que ningun SO de ahora es compatible con la cuantica, y menos un microprocesador. Ninguno (por ahora) esta capacitado para operar sobre un sistema cuantico . La razon es que al leer toda la informacion de los Qubits, se desestabilizaria el sistema. Bue, entonces windows...

d. La competencia de las grandes empresas y el futuro capitalismo
~~~~~~

Sin duda uno de los futuros problemas seran los .. (respeto las reglas SET) que quieren lograr el primer ordenador cuantico para monopolizar todo y para llenarse de guita como hizo Microsoft (R) en algun tiempo. Tampoco faltaran los que les roben las ideas a otros como hizo Microsoft (R) en algun tiempo. Tampoco faltaran los que larguen un computador cuantico primero, y por apuro a serlo, sea una cagada completamente incompatible y desestabilizada, como hizo Microsoft (R) en algun tiempo. Lamentablemente es imposible. Sea IBM o alguna empresa China (que esta un escalon mas adelantada en esto) se robaran todo el capital... Y la historia se volvera a repetir, nosotros peleando por esa causa y el diablo peleando por su poder. Tambien seria bueno que algun grupo o empresa con la ideologia hacker logre la primera computadora.
Ahora las grandes empresas estan compitiendo por eso. Por supuesto que si no seria gracias a los cientificos fisicos, nadie sabria nada de la cuantica, asique creo que el mayor merito hay que darselos a ellos.

Un grupo de japoneses ya lograron algo que permitiria hacer una computadora cuantica de materiales solidos. Evitando hacerla de materiales gaseosos que afectan en muchos aspectos.

En el '94 el Matematico Peter W. Shor desarrollo un algoritmo de factorizacion que permitia comenzar a estudiar la ruptura de sistemas criptograficos

En el '99 los Matematicos Peter W. Shor y Andrew M. Steane ingeniaron un sistema que seria el comienzo para, en un futuro, resolver el problema del primitivo uso de los operadores logicos.

Tambien en el '99 el grupo japones implemento un operador logico de los que se necesitarian para construir un computador cuantico.

No tuve tiempo de revolver mas entre la Red para buscar otros avances, pero basicamente se puede decir que ahora se esta tratando de resolver el tema de como hacer una computadora con materiales solidos y NO gaseosos, se estan investigando los operadores logicos y probando distintos sistemas de
~~~~~~

factorizacion y de encriptacion.

e. Segun la cuantica: La realidad no es lo real
~~~~~~

Hay muchas teorias que ya van mas alla de los estudios computacionales. Esto es mas que todo visto por los fisicos.
En un estudio realizado en el Instituto Nacional de Estandares y Tecnologia de Yankilandia observaron que un atomo puede estar en dos lugares al mismo tiempo. Aqui entra en juego la Tele transportacion, que si leiste el articulo hasta aqui tenes que saber de que hablo. Quizas llego el siglo en el que se hacen realidad los sue~os de los directores de peliculas de ciencia-ficcion.
Pero la teoria mas sorprendente es de la que habla mucha gente que dice que lo que nosotros vemos no es mas que una asimilacion de nuestra mente pero que en realidad no es lo real, sino un sinnumero de juegos de la fisica con nuestra mente.

O sea que no es real, es lo que nosotros vemos, con lo que la mente juega para hacerte ver otra cosa. Un ejemplo MUY cierto y que puede afirmar esta teoria el de la revista Muy interesante de Julio del 2000 Nø 230 (facilitado por blackngel) que dice:
"Cuando nosotros asimilamos que la bola blanca de billar golpea a la otra, en REALIDAD lo que pasa es que los campos electricos de los electrones de los atomos se repelen (como alguna vez en la escuela estudiaste la estructura atomica). Si no fuera por esto una pasaria facilmente a traves de la otra. Entonces hasta donde llega el limite de lo que se puede hacer con el mundo atomico?.
REVISTA MUY INTERESANTE : "Segun la cuantica la realidad misma "deja de existir", es solo una ilusion. No vemos las cosas en si mismas, sino ASPECTOS de lo que son. Al mirar modificamos el mundo."

Como siempre y siempre dije, la mente juega con nosotros dandonos aspectos de las cosas, no realidades. La mente no nos muestra los atomos y la energia de una colision de coches, sino solo los objetos en si. El que piense que estoy delirando que largue el articulo. Para los creyentes y conocedores sobre este tema de la energia otra pregunta:
¨Por que no vemos el aura de las personas?.

4. Mi conclusion
~~~~~~~~~~~~~~

Las computadoras cuanticas van a revolucionar el mundo de las computadoras. el software y el hardware seran CASI perfectos, destacando los procesadores. Por supuesto las computadoras seran mil veces mas rapidas, los criptogramas mas complejos, asi como los desencriptadores. La seguridad avanzara paralelamente con la inseguridad. El sistema cuantico nos permitira seguir evolucionando las computadoras, transpasando la frontera subatomica que por las leyes fisicas de Newton no "pueden" ser transpasadas. Junto con toda esta revolucion vendran los capitalistas contra los que habra que sufrir, como siempre. Otro de los descubrimientos seran los avances a muchas de las preguntas a lo paranormal.

5. Fin
~~~

a. Agradecimientos
~~~~~~~~~~~~~~~~

Quiero agradecer a la ezine SET que me permite largar este articulo y les doy animo a SET para que sigan sacando la ezine con la mayor cantidad de info y lo mas rapido posible. Tambien quiero decirles que voy a seguir mandando articulos de lo que sea.
Tambien quiero agradecer a blackngel que me facilito la informacion para

escribir el tema de "Segun la cuantica: Lo real no es lo real".

Tambien para la SET y para todos los que escriben en esta revista quiero darles una opinion. No se porque si !estamos en argentina o hablamos espa~ol tenemos que andar llamandoles "by pirulo" en vez de "por pirulo". A ver si nos ponemos las pilas y desde aca tratamos que los de afuera no nos dominen. No lleguemos a extremos con las palabras clasicas como "ezine" o Hacker, lammer..., por supuesto. Est no se lo digo tanto por SET, sino por los que mandan articulos.
Esto que les hago es una sugerencia, consejo, exigencia y protesta.

b. Donde encontrar mas informacion
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Escasea si hablamos de paginas especificas, pero aca les mando algunos links:

http://geocities.com/fisica_que/ ’ Muy buena pagina con un indice completo
http://usuarios.iponet.es/ddt/fisica.htm
http://mx.dir.yahoo.com/ciencia_y_tecnologia/fisica/Fisica_Cuantica/
http://www.uco.es/~fa1fepai/fcapli/fcapli.html

... Si hay algun link roto, es que llegaron tarde. Entonces investiguen.

c. Contactame
   ~~~~~~~~~~

Quiero brindarles mi e-mail para que me manden cualquier puteada, consejo, preguntas, reclamo, por el ejercicio, elogio, para mas informacion, errores, sugerencia o lo que se te plazca.

serv_tecnico_inscripcio@hotmail.com
o
lucianobello@hotmail.com
(YY el que tiene algun problema con este email que me lo diga en la cara!!)

Hasta el proximo articulo... []Hacke_R3Ng0[]

EOF

```
-[ 0x10 ]--------------------------------------------------------------
-[ COMienzos ]---------------------------------------------------------
-[ by FCA0000 ]-------------------------------------------------SET-29--
```

En este articulo voy a explicar algo sobre COM

Cuando instale los drivers en Windows para mi telefono Siemens S45 tambien se instalo una aplicacion llamada Data Exchange Software.
Basicamente lo que hace es crear una nueva carpeta llamada "Mobile" bajo la rama "My Computer" en el explorador de archivos.
Al abrir o expandir dicha carpeta, se establece la conexion con el telefono a traves del puerto seleccionado y se accede a la memoria FLEX del movil, que es un sistema de archivos.
Entonces se pueden copiar archivos o crear nuevas carpetas. La manera de hacerlo es la tipica de Windows: seleccionar los archivos del disco , copiar, seleccionar el directorio en el movil, y pegar. Tambien se puede usar el metodo de pinchar y arrastrar (Drag&Drop).

Yo he hecho una aplicacion en el movil, desarrollada en WML/WMLScript, que permite leer un texto en el movil. Para ello tengo un programa en el ordenador que parte el fichero con el texto en varios trozos de 1 Kb. Luego los tansforma en ficheros WML que se pueden visualizar en el movil con el navegador, que entiende lenguaje WML (subconjunto de HTML) y enlaza cada trozo con el siguiente. Una vez que tengo los trozos, debo transferirlos desde el ordenador al movil. Esto lo hago manualmente con el Data Exchange Software (en adelante, DES). Mi objetivo es automatizar este paso.

Una gran diferencias entre Windows y UNIX es que Windows intenta ser un entorno totalmente grafico para hacer mas comprensible el uso, mientras que UNIX usa comandos en linea para que las tareas puedan ser invocadas desde otros programas.
Pero una de las cosas mas frustrantes de Windows es que no es facil hacer que un programa haga algo sin usar el interface grafico: por ejemplo, todos los ficheros tienen unas propiedades accesibles con el boton derecho del raton. Alli se pueden ver cosas como la ultima fecha de acceso, la version, o un resumen con el titulo, tema, categoria... La pregunta es: Alguien sabe como extraer esa informacion para usarla en un informe?

Lo mismo sucede en este caso: Cada vez que quiero copiar los archivos tengo que marcarlos, abrir la carpeta del DES, seleccionar el directorio, y pegar.
Seria estupendo si pudiera hacer
copy C:\ficheros*.WMLC \Mobile
Pero se queja porque \Mobile no es un subdirectorio en C:
Tambien he intentado
copy C:\ficheros*.WMLC \\.\Mobile , esto es, usar el nombre completo, pero tampoco funciona.
Y esto es lo malo: puedo ver la carpeta, pero no puedo usarla a no ser que use el raton.

(Un pequenio inciso: si el movil esta conectado por infrarrojos, se hace
irftp C:\ficheros\aaa.WMLC C:\ficheros\aab.WMLC C:\ficheros\aac.WMLC ...
pero el puerto infrarrojos esta en la parte trasera del ordenador y es de mas dificil acceso. Ademas no se puede hacer
irftp C:\ficheros\a*.WMLC
Y tampoco es el objetivo de este articulo, hombre)

Para empezar, ademas de crear la carpeta de acceso al movil en el explorador, se ha intalado una aplicacion en el menu
Inicio->Programas->Siemens AG->Data Exchange Software
Uno de ellos es la ayuda, que apunta a
E:\Program Files\Siemens AG\Data Exchange Software\des-help.hlp
Otro de ellos se llama Mobile, y es un enlace con las propiedades:

```
Tipo de archivo: Shortcut
Descripcion: Mobile
Location: E:\Documents and Settings\All Users\Start Menu\...
                            ...\Programs\Siemens AG\Data Exchange Software
Target Type: Mobile
Target location: My Computer
Target: Mobile

Asi que empezamos por el directorio
E:\Program Files\Siemens AG\Data Exchange Software\
Alli encontramos el fichero de ayuda, el Shortcut anterior, ademas de
la aplicacion DESSearchApp.exe, que ocupa 40 Kb
y la libreria DESShellExt.dll, que ocupa 864 Kb

El fichero DESShellExt.dll tiene:
Version: 1.0.15.1
Descripcion: DESShellExt Module
Version del producto: 2, 2, 2, 0

Arranco mi desensamblador favorito, URSoft W32Dasm , y miro DESShellExt.dll
Es una libreria dinamica, pero lo primero que sorprende es que tiene un menu
bastante completo con todo lo que se puede hacer:
-Ver ficheros como iconos grndes, pequenios, lista, detallado
-Ordenar por nombre, tipo, tamanio, automaticamente
-Copiar, pegar, renombrar, borrar, propiedades
-Comandos del movil: Formatear, crear directorios estandar, espacio libre
Ah, este menu esta en catalan. Vaya usted a saber porque.
Pero yo lo que voy buscando son los puntos de entrada: Funciones exportadas

 Addr:100129E4 Ord:   1 (0001h) Name: DllCanUnloadNow
 Addr:100129F0 Ord:   2 (0002h) Name: DllGetClassObject
 Addr:10012ADC Ord:   3 (0003h) Name: DllRegisterServer
 Addr:10012ECF Ord:   4 (0004h) Name: DllUnregisterServer
 Addr:1001301F Ord:   5 (0005h) Name: StartSearch
 Addr:100467D0 Ord:   6 (0006h) Name: ?hallole@@3HA

Bien, asi que es una DLL que se puede cargar, descargar, y que es capaz de
registrar un servidor. Tambien hay algo para OLE. Luego veremos mas sobre esto.

Cargando el programa Dependency Walker, que viene con el Kit de desarrollo
de Microsoft, por ejemplo con Visual C++ 6.0
vemos que efectivamente exporta esas funciones pero ?hallole@@3HA tiene
Hint=0x0000 en vez de 0x0006. O sea que se enlaza con doble direccion.

Vamos a seguir recabando informacion. Cargo otras herramientas de espionaje:
-API monitor (http://www.rohitab.com/apimonitor)
-File Monitor
-Registry Monitor
-Process Explorer
(estas ultimas disponibles en www.sysinternals.com)

Cuando pinchamos el icono de Mobile en el explorador, File Monitor nos dice
que se acceden a los siguientes archivos:
DESShellExt.dll
E:\Program Files\Common Files\Siemens AG Shared\DESServer.exe

El fichero DESServer.exe tiene
Version: 1.0.15.1
Descripcion: DESServer Module
Version del producto: 2, 2, 2, 0

Y regmon nos dice que se acceden a las siguientes claves del registro:
```

```
{ED65AB21-B24F-11d3-BA80-00C0CA16AA37} -> Mobile
{ED65AB22-B24F-11d3-BA80-00C0CA16AA37} -> Mobile ContextMenuHandler
{ED65AB23-B24F-11d3-BA80-00C0CA16AA37} -> Mobile PropertySheetHandler
{A78241A1-AD8A-11D3-A271-00105A3B325A} -> DESServer Type Library
{A78241A2-AD8A-11D3-A271-00105A3B325A} -> DESServer
{D73D7EA7-AD8A-11D3-A271-00105A3B325A}
{D73D7EA8-AD8A-11D3-A271-00105A3B325A} -> DESMobilePhone Class
{8DAB6603-AD9C-11D3-A271-00105A3B325A}
{5D764180-EEEC-11d3-BA80-00C0CA16AA37} -> Siemens MMC Search Handler
{01995C2B-2E4C-11D5-999D-0050DA3F471F}
{50C6899A-2D8A-11D5-999C-0050DA3F471F}
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FindExtensions\...
  ...\Static\SiemensMMCSearchHandler\0\DefaultIcon
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\...
  ...\Shell Extensions\Approved
HKEY_CURRENT_USER\Software\Siemens\Des\


Ademas, con el administador de tareas vemos que hay un nuevo proceso
llamado DESServer.exe
Arrancamos la aplicacion Process Explorer y vemos que ha sido arrancado por
svchost.exe
Eso quiere decir que no depende de explorer.exe, sino que mas o menos es
equivalente a un demonio.
La linea de comando ha sido
DESServer.exe -Embedding
Lo importante es que DESServer.exe tambien ha arrancado DESShellExt.dll
Los enlaces abiertos por DESServer son:
\RPC Control\OLE1f , que es un extremo de una comunicacion entre procesos.
\Device\KsecDD , que es tambien un extremo de una comunicacion, o un driver.
HKCU\SOFTWARE\Siemens\DES\MMJ , que es una clave del registro.

Estoy intrigado con esta ultima clave, ya que no existe.
Con mi editor hexadecimal abro DESShellExt.dll y veo que contiene la cadena
HKCU\SOFTWARE\Siemens\DES\MMJ
y cerca de ella, las palabras "FileName" y "Flags"
Asi que usando Regedit voy a la clave HKCU\SOFTWARE\Siemens\DES
y creo la subclave HKCU\SOFTWARE\Siemens\DES\MMJ
Ahora con RegMon veo que accede a la clave Flags, asi que la genero.
No conozco el tipo ni el valor, asi que empiezo por tipo=REG_BINARY y valor=1
Entonces noto que busca el valor de la clave "FileName" que decido crear
como tipo=REG_SZ y valor="aaaaaa"
No pasa nada, y decido cambiar el valor de Flags="FF"
Que suerte! Ahora se ha generado el fichero aa_1aaaa que miro rapidamente:

6 Entering DESServer WinMain
6 Entering GetComPorts
6 Port Mask: 0x00000b
6 Beginning updating Registry.
6 Registry Updated
6 Registring message filter.
8 Entering OBEXCom::OBEXCom()
8 Entering OBEXCom::Connect(WCHAR *port,WCHAR **bmpslots,WCHAR **midslots)
5 CSecurityLayer::Connect       -> AT+CGMM
5 CSecurityLayer::Connect       -> BFB Ping
5 CSecurityLayer::ObexOp:       Send Obex Block and go to WAIT STATE
8 Entering OBEXCom::GetAttribMMC(DWORD* dwData, BYTE byAttrib)
8  83 00 09 4c 00 06 32 01 01
5 CSecurityLayer::Disconnect    Entering<<<<<<<<<<<<<<<<<<<<
5 Switching back to GIPSY: "AT^SQWE=2" was sent.
3 CSerialInterface::CloseConnection     Serial interface closed
4 CSecurityLayer::~CSecurityLayer()
```

2 CSerialInterface::~CSerialInterface

He cortado un monton de lineas pero espero que se entienda que se establece una conexion OBEX con los comandos AT+CGMM y se encapsulan en protocolo BFB. Se ejecuta el comando GetAttribMMC y al final se cierra la conexion,.
La traza es 50 veces mas grande que esto, pero esta orientada a los datos que van por el puerto serie. A mi no me interesa esto, ya que mi objetivo es dejar que DESServer haga ese trabajo por mi.

Por supuesto, todas las cadenas anteriores ("Entering DESServer WinMain", ...) estan dentro de DESServer.exe , asi que estoy bastante seguro que no me estoy dejando ningun programa por analizar. Y cada vez me doy mas cuenta de que DESShellExt.dll no me va a servir de mucho.

Si hubiera una manera de generar otro fichero con la traza
entre "Entering DESServer WinMain" y "Entering GetComPorts" me ayudaria mucho.
Pero por mas que miro en el registro no encuentro nada.
Seguimos por otro camino: sabemos que alguien llama al programa DESServer.exe con argumentos -Embedding asi que hago una copia de seguridad, copio el Notepad.exe y le doy el nombre DESServer.exe
A continuacion pincho el movil, y efectivamente arranca el Notepad.exe , con lo cual estoy seguro de que estoy en la pista correcta. No solo eso, sino que notepad se queja de que no encuentra el archivo -Embedding . Bien, los argumentos tambien son correctos.
O sea, que puedo llamar a DESServer.exe cuando me apetezca, con lo cual se arrancara el servidor. Lo curioso es que no encuentro la palabra "Embedding" en ninguno de los ficheros. Posiblemente sea Windows quien lo manda como argumento pero DESServer.exe no hace nada con el.

Volvemos sobre nuestros pasos hasta donde vemos que es importante la clave CLSID
{ED65AB21-B24F-11d3-BA80-00C0CA16AA37}
Un sitio donde se encuentra es
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\...
   ...\MyComputer\NameSpace\
y el segundo es
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\...
    ...\CurrentVersion\Shell Extensions\Approved

La primera clave nos dice que {ED65AB21-B24F-11d3-BA80-00C0CA16AA37} es una extension de NameSpace de Explorer, identificado como "Mobile" y que depende de "MyComputer"
La segunda nos dice que es una extension de Shell, es decir, que ademas es una extension de Shell.
Segun dice la documentacion del MSDN de Microsoft, un NameSpace es una coleccion de simbolos, tales como claves de bases de datos, archivos, y nombres de directorios. El shell usa un NameSpace jerarquico para organizar los objetos, incluyendo archivos, dispositivos de almacenaje, impresoras, recursos de red, y cualquier cosa que se puede ver usando el Explorer.
El escritorio es la raiz de todos ellos.
Dentro de un NameSpace hay objetos llamados carpetas. Algunas carpetas son directorios, pero no todas. Por ejemplo, el panel de control es una carpeta pero no un directorio.
Dado que hay muchos tipos de carpetas y objetos, cada carpeta es un COM: modelo de objeto de componente OLE. Mas concretamente, cada carpeta implementa el interface IShellFolder .
Espero que a partir de ahora useis correctamente los terminos carpeta y directorio, que, COMo se ha visto, no son lo mismo.

Ah, eso aclara muchas cosas. Por ejemplo, podemos abrir el explorador de archivos y escribir en la barra de direcciones:

```
::{20D04FE0-3AEA-1069-A2D8-08002B30309D}\::{ED65AB21-B24F-11d3-BA80...
     ...-00C0CA16AA37}
```

porque {20D04FE0-3AEA-1069-A2D8-08002B30309D} es "MyComputer", y el
otro CLSID es el del objeto "Mobile"
O lo que es lo mismo: abrir explorer.exe y escribir en la barra de
direcciones Mobile\Misc

Similarmente podemos abrir una ventana de comandos y escribir

```
explorer.exe /e,/root,::{20D04FE0-3AEA-1069-A2D8-08002B30309D}\::{450D8FBA...
    ...-AD25-11D0-98A8-0800361B1103}
```

y abre la carpeta "MyDocuments"

y ::{6B19FEC2-A45B-11CF-9045-00A0C9039735} abre la
carpeta "Registered ActiveX Controls"
tambien ::{D6277990-4C6A-11CF-8D87-00AA0060F5BF} abre "Scheduled Tasks"

Incluso mas: en vez de arrancar el explorador de archivos, ejecutamos el
explorador de internet:

```
iexplore "::{20D04FE0-3AEA-1069-A2D8-08002B30309D}\::{ED65AB21...
   ...-B24F-11d3-BA80-00C0CA16AA37}"
```

O abrimos Internet Explorer y en la barra de direccion escribimos

```
::{20D04FE0-3AEA-1069-A2D8-08002B30309D}\::{ED65AB21...
   ...-B24F-11d3-BA80-00C0CA16AA37}\Misc
```

y abre ese directorio. Claro que tambien funciona con   Mobile\Misc

Lamentablemente no funciona abrir un directorio desde la linea de comandos,
desconozco la razon.

Bueno, pues podemos abrir el IShellFolder "Mobile". Ahora que?
Pues podemos llamar a IShellFolder::EnumObjects para que nos diga los
ficheros que hay dentro. Esto se encargara de llamar al servidor COM (que no
es otro que DESServer.exe) y la funcion adecuada que saca el listado completo.

Segun la documentacion, podemos crear un directorio en cualquier disco, con
el nombre de un directorio en el namespace (esto es, en el movil) y extension
igual al CLSID. O sea:

```
mkdir c:\miMovil.{ED65AB21-B24F-11d3-BA80-00C0CA16AA37}
```

Efectivamente cuando hacemos doble-click, se abre el directorio en el movil.
Pero solo desde el Explorer. No funciona desde la linea de comandos !
De nuevo, dice que no hay ficheros cuando se hace

```
DIR c:\miMovil.{ED65AB21-B24F-11d3-BA80-00C0CA16AA37}
```

Mas sorprendente (y esto es un buen truco para esconder carpetas) es hacer

```
e:\>mkdir prueba.{450D8FBA-AD25-11D0-98A8-0800361B1103}
e:\>mkdir prueba.{6B19FEC2-A45B-11CF-9045-00A0C9039735}
```

Lo cual creara 2 directorios con el mismo nombre! porque Windows oculta
la extension cuando se trata de un CLSID

Lo que hemos aprendido es que vamos a tener que acceder usando las funciones
de COM. Pero cuales son los metodos que podemos llamar en DESServer.exe ?
Para saberlo usamos la aplicacion OLEview, tambien conocida
como "OLE/COM Object Viewer" que viene incluida con la instalacion
de Miscrosoft Visual Studio 6.0 o con el Windows SDK.
Podemos ver todos los objetos COM disponibles y buscar el nuestro, pero
es mas sencillo usar el menu "View TypeLib" y abrir el fichero DESServer.exe
Obtenemos algo asi como:

```
// Generated .IDL file (by the OLE/COM Object Viewer)
//
// typelib filename: DESServer.exe

[
  uuid(A78241A1-AD8A-11D3-A271-00105A3B325A),
  version(1.0),
  helpstring("DESServer Type Library")
]
library DESServer
{
    // TLib : OLE Automation : {00020430-0000-0000-C000-000000000046}
    importlib("stdole2.tlb");

    // Forward declare all types defined in this typelib
    interface IDESVolume;
    interface IDESGeneral;

    typedef enum {
        stBMP = 0,
        stMID = 1
    } SlotTypeConstants;

    [ odl,
      uuid(D73D7EA7-AD8A-11D3-A271-00105A3B325A),
      helpstring("IDESVolume Interface"),
      dual,
      oleautomation ]
    interface IDESVolume : IDispatch {
        [id(0x00000005), helpstring("Creates the standard directory.")]
        HRESULT CreateStdDirs();
        [id(0x00000006), helpstring("Deletes an existing file.")]
        HRESULT Delete([in] BSTR path);
        [id(0x00000010), helpstring("Creates a new file.")]
        HRESULT PutFile(
                        [in] BSTR path,
                        [in] IUnknown* content,
                        [in] long attributes);
   };

    [ odl,
      uuid(CA2AF16E-A466-41E8-A32A-FF86D6D955A6),
      helpstring("IDESGeneralInterface"),
      dual,
      oleautomation ]
    interface IDESGeneral : IUnknown {
        [id(0x00000016), helpstring("Returns the Mobile Model connected.")]
        HRESULT GetMobileModel([out, retval] BSTR* pbstrModel);
    };

    [
      uuid(D73D7EA8-AD8A-11D3-A271-00105A3B325A),
      helpstring("DESMobilePhone Class")
    ]
    coclass DESMobilePhone {
        [default] interface IDESVolume;
        interface IDESGeneral;
    };
};
```

```
Esto es un fichero de tipo IDL. He cortado muchas de las lineas para
hacerlo mas legible.
Vemos que se estructura en una libreria
con id={A78241A1-AD8A-11D3-A271-00105A3B325A}
con alias "DESServer Type Library"
que contiene un objeto
{D73D7EA8-AD8A-11D3-A271-00105A3B325A}, conocido como "DESMobilePhone"
que a su vez contiene 2 interfaces:
{D73D7EA7-AD8A-11D3-A271-00105A3B325A}, conocido como "IDESVolume"
  que implementa las funciones CreateStdDirs, Delete, PutFile
{CA2AF16E-A466-41E8-A32A-FF86D6D955A6}, conocido como "IDESGeneral"
  que implementa las funciones GetMobileModel (y otras que no he listado)

Hay otro objeto de tipo Application
{A78241A2-AD8A-11D3-A271-00105A3B325A}, conocido como "IDESServer"
Tras mucho investigar en COM, he llegado a la conclusion de que para usar esto
en Visual Basic, hay que incluir en el menu Proyecto->Referencias la
libreria para "DESServer Type Library". Esto permite usar las funciones
incluidas en los interfaces.
Un programa basico es algo asi como:

Private Sub Form_Load()
Dim x As Object
Set x = CreateObject("DESServer.DESMobilePhone")
Call x.CreateStdDirs
Call x.Delete("prueba.txt")
End Sub


Como veis, lo primero es crear una referencia al objeto.
Despues, cuando se ejecuta la linea para crear el objeto, esto mira en el
registro, crea la libreria DESServer y el objeto DESMobilePhone
Por defecto esto apunta al interface IDESVolume.
Ahora podemos llamar a cualquiera de sus metodos, por ejemplo CreateStdDirs.
Tambien es facil llamar a una funcion con argumentos.
Pero yo lo que pretendo es llamar a la funcion
PutFile( [in] BSTR path, [in] IUnknown* content, [in] long attributes);
Lamentablemente en Visual Basic no se como pasar un puntero a IUnknown.

Asi que pasamos a otro lenguaje mas manejable: C++
Usando Visual C++ 6.0 lo primero que tengo que hacer es convertir el
fichero IDL en algo que el compilador entienda:
c:\midl /h DESServer.h DESServer.IDL

que genera una cabecera DESServer.h (recortada) :

    MIDL_INTERFACE("D73D7EA7-AD8A-11D3-A271-00105A3B325A")
    IDESVolume : public IDispatch
    {
    public:
        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE Delete(
            /* [in] */ BSTR path) = 0;
        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE PutFile(
            /* [in] */ BSTR path,
            /* [in] */ IUnknown __RPC_FAR *content,
            /* [in] */ long attributes) = 0;
    };

y tambien un fichero DESServer.TLB, y un fichero DESServer_i.c con
las constantes adecuadas:
const IID LIBID_DESServer = {0xA78241A1,0xAD8A,0x11D3,
                                {0xA2,0x71,0x00,0x10,0x5A,0x3B,0x32,0x5A}};
```

```
const IID IID_IDESVolume = {0xD73D7EA7,0xAD8A,0x11D3,
                                    {0xA2,0x71,0x00,0x10,0x5A,0x3B,0x32,0x5A}};
const IID IID_IDESGeneral = {0xCA2AF16E,0xA466,0x41E8,
                                    {0xA3,0x2A,0xFF,0x86,0xD6,0xD9,0x55,0xA6}};
const CLSID CLSID_DESMobilePhone = {0xD73D7EA8,0xAD8A,0x11D3,
                                    {0xA2,0x71,0x00,0x10,0x5A,0x3B,0x32,0x5A}};
que luego nos seran muy utiles

Asi que el progama es algo asi:
#include "DESServer.h"

....
IUnknown *iu=NULL;
IDESVolume FAR* iu2 = (IDESVolume FAR*)NULL;
HRESULT hr = CoInitialize(NULL);
hr=CoCreateInstance(CLSID_DESMobilePhone, NULL, CLSCTX_ALL,
                                        IID_IUnknown, (void **)&iu);
// usamos CLSID_DESMobilePhone porque automaticamente
// instancia LIBID_DES Server como padre
hr = iu->QueryInterface(IID_IDESVolume, (void **)&iu2);
hr = iu2->Delete(L"prueba.txt");

// Facil, eh? Ahora vamos con el fichero: necesitamos un IID_IPersistFile que
// usa un archivo del disco. Y el disco tambien es hijo del escritorio (Shell)
IPersistFile *iu3;
IShellLink *psl;
hr = CoCreateInstance (CLSID_ShellLink, NULL, CLSCTX_INPROC_SERVER,
                                         IID_IShellLink, (void **)&psl);
hr = psl->QueryInterface (IID_IPersistFile, (void **)&iu3);
iu3->Load(L"prueba.txt",STGM_READ); // archivo original
hr = iu2->PutFile(L"prueba.txt",iu3,0); // archivo en el movil


Mas util es poder invocar a un metodo segun un numero, no segun el nombre.
En el IDL hemos visto que la funcion Delete tiene el indice 0x00000006
Para ello hay que usar la funcion de mas bajo nivel  IUnknown::Invoke
Asi que podemos hacer:

DISPID dispid;
OLECHAR FAR* szMember = L"Delete";
hr = iu2->GetIDsOfNames(IID_NULL, &szMember, 1,
                LOCALE_USER_DEFAULT, &dispid);
// estas dos lineas anteriores hacen que dispid=0x00000006
// No son necesarias porque precisamente ya sabemos que Delete=0x00000006
VARIANT vtResult;
EXCEPINFO  excepinfo;
UINT    argerr;
VARIANTARG  rgvt[1]; // Solo hay un argumento
V_VT(&vtResult) = VT_HRESULT;
V_VT(&rgvt[0]) = VT_BSTR;
V_BSTR(&rgvt[0]) = L"prueba.txt";
DISPPARAMS  dispparams = { rgvt, NULL, 1, 0 };
hr = iu2->Invoke(
        dispid,
        IID_NULL,
        LOCALE_USER_DEFAULT,
        DISPATCH_METHOD,
        &dispparams, &vtResult, &excepinfo, &argerr);

Si algo falla, el parametro  excepinfo  contiene informacion extra y
tambien  hr  puede devolver (entre otros) :
DISP_E_BADPARAMCOUNT = incorrecto numero de argumentos
```

```
DISP_E_BADVARTYPE    = uno de los argumentos no es del tipo correcto.
                           el valor argerr deberia decir cual de ellos es.
DISP_E_EXCEPTION     = el metodo ha lanzado la excepcion. Por ejemplo, puede
                          que otra variable necesite ser definida anteriormente.
DISP_E_TYPEMISMATCH  = uno de los parametros no es del tipo correcto y tampoco
                           se puede encontrar una conversion adecuada.

Estos valores estan definidos en WINERROR.H

O sea, que comodamente podemos usar cualquier funcion que este definida en un
objeto COM. Asi podemos usar practicamente casi cualquier funcionalidad de
cualquier libreria. Ejemplos de esto son:
-un programa que imprima la primera hoja de una serie de documentos PDF
-un programa que aniada usuarios al sistema, a partir de un fichero. Hace
  mucho que me gustaria tener un programa que cree un grupo para cada
  usuario. Ahora puedo hacerlo.
-llamar a Paint para que reduzca el tamanio de una serie de ficheros BMP
-filtrar los eventos que van al Event Viewer para automatizar alarmas
-mandar una serie de calculos para que la Calculadora los evalue. Con esto
  se podria implementar facilmente el programa   bc   de UNIX

En otras palabras: Windows ha creado una casa con muchas habitaciones y solo
hay que llamar a la puerta para conseguir acceso mas completo al sistema.


En este caso podria haber utilizado otra tecnica tambien basada en COM.
Como todos sabreis, en Windows casi todas las aplicaciones tienen un menu
llamado "Edicion" para copiar datos en el portapapeles y luego pegarlos
en otro sitio, quizas en otra aplicacion.
Existen varios formatos, entre ellos CF_TEXT, CF_TIFF, CF_WAVE, ... Esto
permite que un sonido se pueda pegar desde una aplicacion
para encajarlo en otra, y sera Windows quien los controle.
Es posible registrar nuevos formatos, enlazandolos con una aplicacion
desarrollada por nosotros. Cuando se copie/corte/pegue, nuestra libreria
sera llamada para operar con los datos. Asi se puede, por ejemplo, incrustar
un fichero ZIP dentro de cualquier aplicacion, ya sea un mensaje de correo
o un documento Word.
Hay 2 aplicaciones muy utiles para esto: el visor del portapapeles clipbrd.exe
y el IDataObject Viewer.
Si seleccionamos un trozo de texto vemos en el IDataObject que el dato es
posible interpretarlo (entre otros) como de tipo CF_TEXT, o sea, que es
simplemente una serie de letras.
Si seleccionamos un trozo de HTML vamos que el formato es ademas interpretable
como "HTML Format".
Si seleccionamos un archivo, el tipo es "FileName" e incluso
tambien "Shell IDList Array". ?Adonde mos lleva esto? Pues primero a un poco
de investigacion en el MSDN, a vueltas con las extensiones del Shell, y ver
que toda la gracia esta en torno a:
OleGetClipboard, OleCreateFromData , IDataObject, CFSTR_SHELLIDLIST,
FileGroupDescriptor, y InvokeCommand.
Y en Visual Basic es realmente sencillo:

Declare Function OpenClipboard Lib "User32" (ByVal hwnd As Long) As Long
Declare Function GetClipboardData Lib "User32" (ByVal wFormat As Long) As Long
Declare Function GlobalLock Lib "kernel32" (ByVal hMem As Long) As Long


Const CF_TEXT = &H07
OpenClipboard(0&)
hClipMemory = GetClipboardData(CF_OEMTEXT)
lpClipMemory = GlobalLock(hClipMemory)
Dim c2 as Object
```

```
Set c2 = CreateObject("DESServer.DESMobilePhone")
c2.Paste lpClipMemory
```

Solo falta el paso inicial de meter los datos en el portapapeles, pero eso lo tengo hecho en C++ y ocupa muchas mas lineas. (?Para que demonios se necesita un puntero a una ventana cuando en realidad estoy accediendo a un servidor?)

Si buscas DataFormats en el registro te puedes hacer una idea de cuantas aplicaciones usan un formato propio para poder usar el portapapeles.

En este caso hemos usado una tecnica similar: hay funciones en las liberias User32.dll y kernel32.dll que podemos usar a nuestra conveniencia si sabemos los parametros adecuados.
Gracias a MS, las dll que vienen con Windows estan documentadas y se pueden usar sin mas problemas que saber los tipos de los argumentos.

Un ejemplo vale mas que 1000 palabras:
Supongamos que queremos llamar a la funcion NtQueryInformationFile que esta en ntdll.dll
Lo primero es saber sus argumentos; como es interna al kernel de NT, la primera fuente de informacion es el Win32 SDK, pero alli no esta documentada, asi que consultamos el Windows NT DDK. Alli esta definida como:

```
NTSTATUS (__stdcall *NtQueryInformationFile)(
    IN HANDLE FileHandle,
    OUT PIO_STATUS_BLOCK IoStatusBlock,
    OUT PVOID FileInformation,
    IN ULONG Length,
    IN FILE_INFORMATION_CLASS FileInformationClass
    );
```

Los parametros son autoexplicativos, aunque los valores no estan claros y las estructuras tampoco. Pero ahondando en el DDK se encuentra que
-Length puede ser 16384 pero hay que aumentarlo si da STATUS_BUFFER_OVERFLOW
-FileInformationClass es un valor de _FILE_INFORMATION_CLASS con valores, por ejemplo, FileAllInformation=18
-FileInformation es una estructura FILE_INFORMATION conteniendo:
  -una lista anidada mediante el campo .NextEntry
  -el campo .Name
  -otros muchos campos, dependiendo del valor usado en FileInformationClass

Apuntamos a la DLL con

```
NtQueryInformationFile = (void *)
                         GetProcAddress( GetModuleHandle(L"ntdll.dll"),
                                         "NtQueryInformationFile"
                                       );
```

Y la llamamos con

```
        status = NtQueryInformationFile( fileHandle, &ioStatusBlock,
                                         fileInformation, length,
                                         fileInformationClass );
```

El resultado se saca con

```
wprintf(L"\r%s:\n", fileInformation->Name );
```

Facil, eh? Claro que en este caso la funcion estaba documentada y hemos podido saber los argumentos.
En caso contrario las unicas tecnicas que se me ocurren son:
-Desensamblar el programa, y en la entrada a la funcion ver cuantos parametros se leen de la pila, y averiguar los tipos.
-Con un trazador de llamadas como APImonitor o PDump32 poner un breakpoint en la entrada a la funcion.

Otra herramienta utilisima para meter las narices en las tripas abiertas de
una aplicacion es el Editor de Recursos. Yo he usado en
Windows Resource Toolkit de Borland y otro llamado Resource Hunter.
Hay otro mas llamado Resource Extractor que viene con GraphicsWorkshop y
tampoco esta mal.
Con este tipo de programas se ven los recursos de una aplicacion: Bitmaps,
Iconos, Dialogos, Strings, Datos Binarios.
Es posible extraer datos, o bien llamarlos para incluirlos en otro programa
nuestro.

Con esto hemos visto un metodo para usar librerias que en un principio no
estan abiertas ni estan disponibles, pero que estan instaladas en mi ordenador.

Un momento: en "mi" ordenador? Pues rectifico: quise decir en "un" ordenador.
Y esto es asi porque existe una extension a COM llamada COM Distribuido: DCOM.
Un cliente en un ordenador puede usar un objeto ejecutandose en otro ordenador.
Funciona igual, pero el cliente necesita especificar el servidor en el que
desea ejecutar el proceso, asi como definir las credenciales.
Se usa la aplicacion Dcomcnfg para definir los niveles de seguridad del
servidor y del cliente. Notar que tambien se configura el cliente: por
ejemplo puedo hacer que un cliente que intente acceder al servicio DESServer
se conecte a otro ordenador. No hay que interpretar esto como un vinculo
encadenado, sino mas bien que el cliente le pregunta a Dcomcnfg donde
puede encontrar el servicio, y este se lo dice.

Normalmente las aplicaciones usan el nivel de seguridad definido globalmente
para todo el sistema, aunque es posible cambiarlo.
Estos parametros por defecto en Windows2000 Server indican que:
-DCOM esta activado
-los servicios de COM para Internet estan desactivados
-el nivel de autentificacion es Connect. Esto quiere decir que la
  verificacion de seguridad se produce solamente en la primera conexion.
-el nivel de impersonalizacion es Identify. Esto significa que el
  servidor _puede_ verificar la identidad del cliente.
-los permisos de acceso indican que, si la aplicacion no lo define, se heredan
  los de Administrador
-los permisos de ejecucion indican que todos los usuarios pueden ejecutar.
-los permisos de configuracion indican que solo el creador, y
  superusuarios, pueden cambiar la configuracion. Pero todos pueden leerla.
-no se filtran puertos.

En otras palabras: cualquiera que pueda conectarse a la maquina puede
arrancar un objeto.
El caso de DESServer tiene parametros normales que no creo que afecten a
la seguridad, pero no siempre es asi:
Por ejemplo, el servicio {000C101C-0000-0000-C000-000000000046}, tambien
conocido como "MSi Install Server", tiene que la identidad asumida
es "Cuenta del Sistema", o sea, que cualquier usuario que consiga arrancarla
tendra permisos de Sistema.
Lo mismo pasa con "Windows Management Instrumentation", "TIntSrv", "NtmsSvc",
"Microsoft WBEM Server", y otros. Por lo menos, todos ellos son
servicios, asi que de todas maneras se pueden detener.
Esto es asi porque COM se basa en DCE RPC (Remote Procedure Calls) y este
modelo proporciona mecanismos para que sean las aplicaciones las que
hagan sus propios chequeos de seguridad.

La manera de acceder a ellos esta muy bien explicada en el articulo Q180217 y
el MSDN, y basicamente implica llamar a la funcion CoCreateInstanceEx, cuyo
tercer argumento es el nombre del servidor.
Seguramente despues hay que llamar a CoInitializeSecurity que en su forma
mas simple, es:
hr = CoInitializeSecurity(NULL, -1, NULL, NULL, RPC_C_AUTHN_LEVEL_CONNECT,

```
                                RPC_C_IMP_LEVEL_IMPERSONATE, NULL, 0, NULL);
```

La mayor fuente de informacion es el Microsoft Developer Network MSDN, ya que parece que COM es una de las apuestas de MS para la ejecucion distribuida de aplicaciones, sobre todo en el ambito de los servidores.

Como caso practico de hacking, vamos a usar uno de los servicios instalados en Windows2000 para hacer un escalado de privilegios.
Ejecutamos la aplicacion ddeshare.exe
Y vemos que hay dos opciones: DDE shares y Trusted Shares
Elegimos la primera, y aparecen 3: Chat$, CLPBK$, Hearts$
El ultimo es para el juego Hearts (corazones) que al pulsar Propiedades, vemos que los cuadros "Permitir ejecutar aplicacion" y "es Servicio" no estan marcados.
En el segundo (CLPBK$) vemos que es un servicio, pero no se puede ejecutar la aplicacion.
En el primero (Chat$) vemos que se puede ejecutar la aplicacion. No solo eso, sino que la seguridad esta puesta para "Permitir acceso a todos los objetos".
Asi que ejecutamos winchat.exe
Aparece brevemente un mensaje "Starting Net DDE related services" bastante sospechoso, y con el Process Explorer (el Windows Task Manager no muestra procesos del sistema) vemos que hay 2 procesos nuevos:
netdde.exe , tambien conocido como "Network DDE - DDE Communication"
y clipsrv.exe, tambien conocido como "Windows NT DDE Server"

Lo mas gracioso es que ambos han sido ejecutados por el usuario "NT AUTHORITY\SYSTEM". Eso es interesante. Aunque cerremos la aplicacion winchat.exe , estos 2 procesos no mueren.
La manera de matarlos es convertirse en Administrador, y hacer un kill (con las pstools) o desde el mismo Process Explorer.
Matamos los procesos, y ejecutamos de nuevo winchat.exe
Pasa lo mismo que antes: mensaje con "Starting Net DDE related services" y arrancan los 2 servicios.
Ahora usamos en menu conversacion->marcar y escribimos el nombre de nuestro ordenador.
Ojo que no funciona usando "localhost" ni "127.0.0.1". Hay que usar el nombre tal como aparece al hacer
c:\> echo %COMPUTERNAME%
Entonces dice que "el otro ordenador no respondio" pero de todas maneras ha abierto una ventana con winchat.exe para poder hablar. Llamamos a este proceso winchat2, para diferenciarlo de winchat1.
Ahora en winchat1 llamamos otra vez a winchat2, y la ventana de winchat2 parpadea y dice que winchat1 esta llamando. Podemos contestar la llamada para empezar a hablar. A partir de ahora es como un cliente sencillo de IRC.
Por favor, nada de criticas. Chat se desarrollo con Windows 3.11 para grupos, y estaba basado en el antiguo talk de UNIX. Hace 10 anios.

Pero hay un detalle que no puede pasar inadvertido: el Process Explorer muestra 2 procesos winchat.exe :
-uno (winchat1) ejecutado como yo mismo
-otro (winchat2) como "NT AUTHORITY\SYSTEM"
Pulsando doble-click sobre winchat2 vemos que ha sido arrancado por winlogon.exe !!!
Aqui hay un exploit facil:
-hacer una copia de winchat.exe con
  copy e:\winnt\system32\winchat.exe e:\winnt\system32\winchat.old
-hacer otra copia
  copy e:\winnt\system32\winchat.exe e:\winnt\system32\winchat1.exe
-sobreescribir el original con un exploit, ej. el interprete de comandos:
  copy e:\winnt\system32\cmd.exe e:\winnt\system32\winchat.exe

```
-ejecutar  winchat1.exe
-intentar conectar con tu propio ordenador
  esto arrancara el proceso  winchat.exe , que es en realidad  cmd.exe
-en la nueva ventana de comandos, escribir
  echo %USERNAME%
-nos dice exactamente "%USERNAME%" , o sea, que no esta definida esa variable.
-para comprobar que somos SYSTEM, hacer
  cd c:\"System Volume Information"
  Normalmente ese directorio no se puede acceder (ni siquiera Administrador)
  porque pertenece a SYSTEM.

Mucho cuidado con lo que haceis en esta ventana. Ser SYSTEM tiene sus riesgos.

Una cosa que no funciona del todo bien es que cualquier programa ejecutado
desde esta linea de comandos no hereda los privilegios. Lo mas facil es
hacer un programa que haga lo que nosotros queremos hacer (copiar
archivos, instalar un nuevo servicio, ...)
Notar que es el paso importante es que winlogon.exe no ha bajado los
privilegios una vez arrancado winchat.exe , y tambien que el proceso
sobreescrito (cmd.exe) tampoco ha pedido perderlos. En Windows es posible
hacer que un programa padre le quite algunos privilegios a sus hijo. Esto
se hace desde el explorador, pulsando el boton derecho, propiedades,
seguridad, avanzado, "Permitir propagar permisos heredables desde el padre".
Y tambien se puede hacer que tengan diferentes permisos segun el usuario
que lo ha ejecutado. Todo ello lo puede definir el administrador del
sistema, no solo el propio programa. En UNIX esto se hace con el
sticky bit, pero es mas limitado.

Es posible que, al probar este truco, la ventana del cmd.exe no se
abra, sino que se arranca el winchat.exe original. En Windows2000 hay
un mecanismo de control de versiones. Cuando el sistema detecta que
un fichero de sistema no tiene la fecha, el tamanio, o el checksum que
deberia tener, recupera una copia del fichero de la
carpeta e:\winnt\system32\dllcache\ y machaca la que haya
en e:\winnt\system32\ . En este caso hay que borrar la copia original
de e:\winnt\system32\dllcache\  o desactivar este metodo de seguridad.
De hecho este mecanismo de proteccion es un gran invento, si no fuera
porque ocupa 300 Mg. Ademas no solo estan los archivos que se han
instalado, sino todos aquellos que se pueden instalar. Por eso es posible
encontrar aplicaciones en e:\winnt\system32\dllcache\ que no estan instaladas.

Claro que para poder operar con el directorio e:\winnt\system32\ hay que
haber accedido al ordenador previamente, aunque no necesariamente con
privilegios de Administrador.
Por ejemplo, en muchas empresas grandes los empleados tienen instalado
WindowsNT Workstation, pero con permisos limitados. Es comun que no
puedan instalar programas, no puedan acceder al registro, o no puedan
ver el contenido de ciertos directorios en su disco duro. Con este exploit
es posible hacerlo, sin mas requerimiento que tener permiso de
lectura/escritura en e:\winnt\system32\ , lo cual no es raro.

Tambien decir que estos programas usan NetDDE, que es la tecnologia desarrollada
antes de COM, y me parece raro que Microsoft siga usando esos tres programas,
ya que los podian haber modificado para usar DCOM.
La tecnologia DDE usa los puertos tipicos de NetBIOS , que es la misma
usada para SAMBA, el sistema de autentificacion, el de controladores de
dominio, ... y creo recordar que usan los puertos 135, 137 y 138 .
O sea que si el servidor tiene un firewall instalado para esos puertos, no
es posible hacer DDE/COM a traves de Internet.

Para los curiosos, existe una aplicacion llamada DDESpy que permite ver
todo el trafico DDE. Proporciona mucha informacion, pero interpretarla
```

puede ser mas dificil.

Tambien quiero mencionar Active-X. Ya esta. Ya lo he mencionado.

Para finalizar, recalcar que yo tengo instalados mas de 150 objetos COM, con aproximadamente 1500 metodos, y antes de reinventar la rueda prefiero perder un momento en ver si la misma funcionalidad esta disponible en otra aplicacion, y en este caso intentar usarla COModamente.

EOF

-[0x11]--
-[Autentificacion]---
-[by Cemendil]---SET-29--

AUTENTIFICACION DE USUARIOS EN EL S.O. WINDOWS

Autor : Cemendil <cemendil@hotmail.com>
Fecha : 27/01/2004
Version : 0.82

And the walls shall have eyes
And the doors shall have ears
But we'll dance in the dark
And they'll play with our lives

David Bowie, "Slow burn".

---------------------------| Contenidos. |--------------------------

Referencias.

----------| Advertencia |----------

Estimado lector:

Las opiniones expuestas en este articulo pertenecen unicamente al autor del mismo, Cemendil <cemendil@hotmail.com>, y no representan ni reflejan, que yo sepa, la de los editores del e-zine en que ha sido publicado. En cuanto a la libertad del autor para expresar sus opiniones, cf. Constitucion Española, Art. 20, 1.a y 1.d.

Cemendil.

----------| Introduccion |----------

El presente articulo trata sobre los algoritmos empleados en la autentificacion de usuarios en los sistemas operativos Windows. En breve, nos ocuparemos de entender como reduce Windows las frases de acceso ('passphrases' o 'passwords') a una cadena de bytes, que es almacenada en los bien conocidos ficheros de passwords. Este proceso es conocido como 'hashing', y su eficacia se fundamenta en que es teoricamente intratable el obtener un password a partir de su hash.

Los algoritmos de hashing en Windows son dos: en LM-hash y el NT-hash. Ambos son bien conocidos desde hace bastante tiempo. LM-hash es una herencia de IBM, y es universalmente conocido por ser enormemente debil; mas adelante hablaremos acerca de un ataque reciente sobre LM-hash. El NT-hash, por el contrario, es considerado como muy fuerte. En este articulo echaremos un buen vistazo al NT-hash, y compararemos brevemente su intratabilidad con la de otros algoritmos de autentificacion de usuarios, tales como el clasico crypt de UNIX, y el mas moderno crypt de BSD.

Durante este articulo trataremos de abordar dos objetivos. En primer lugar, una cosa es leer acerca de una implementacion de un algoritmo, y otra muy distinta es mojarse el culo, implementandolo y trabajando con el. Pues bien, nos mojaremos el culo, implementando y comprobando ambos metodos de hashing y viendo como funcionan en la realidad. Es tan absurdo pretender que se es biologo tan solo por haber dado un paseo por el zoo como pretender que se entiende un algoritmo por haberlo leido, incluso con cierto detenimiento. Hay que diseccionar y hacer funcionar las cosas para entenderlas.

En segundo lugar, trataremos de demostrar la tesis de que los algoritmos de autentificacion de usuarios en Windows son, cuando menos, irresponsablemente simples. Esto no es nada nuevo en lo que a LM-hash se refiere (la gente habla abiertamente de estupidez, trampa y otras cosas peores), pero tambien se aplica, en menor medida, al NT-hash. Como se vera mas adelante, NT-hash es peor en algunos sentidos que el venerable crypt de UNIX, basado en una modificacion de DES, no digamos ya si se compara con los crypt de FreeBSD o de OpenBSD.

Pensandolo honestamente, parece que los autores de ambos hashes fueron deliberadamente negligentes. Habiendo precedentes tan buenos como el crypt de UNIX, es sospechoso que cosas como NT-hash y LM-hash salieran de la mesa de trabajo de un ingeniero de software

sobrio. Pregunta retorica: ?A quien interesaria que el sistema operativo mas extendido del mundo tenga algoritmos de hashing que, sin ser trivialmente atacables, son sin embargo juguetes comparados con los de los otros SOs? Ya se, ya se, no hay que achacar a la mala intencion lo que pueda ser obra de la estupidez pero, sinceramente, o esto es un milagro de estupidez, o es mala intencion.

?Alguien quiere apostar? O quizas deberia preguntar, ?Alguien quiere correr el riesgo, y aun encima pagar por ello?

----------| Licencia |----------

----------| Parte 1 : Hashing |----------

Antes de entrar con LM-hash y NT-hash, veamos en que consiste el hashing y cuales son las vulnerabilidades mas comunes. A lo largo de la seccion 1.2 haremos comparaciones entre los hashes de Windows y algunos de los hashes mas comunes en UNIX. Sobre los hashes de UNIX hay gran cantidad de informacion, codigo fuente incluido, pero como suele ocurrir con la informacion libre, hay que buscarla por la red. Una visita a la fuente de tu UNIX puede servir de ayuda.

Una buena descripcion del crypt clasico de UNIX, junto a una discusion sobre la autentificacion de usuarios, se encuentra en [1], 10.2.

---| 1.1 De que va la cosa.

Un problema importante en todo sistema informatico es limitar el acceso a las personas autorizadas, evitando que un intruso pueda hacerse pasar por un usuario legitimo. El metodo mas comun consiste en que el sistema comparta un secreto con cada uno de los usuarios legitimos, de manera que se reconozca a los elegidos por esa informacion privilegiada. El secreto suele ser una palabra o frase secreta (en lo sucesivo, 'password'). Naturalmente, el password no es intrinseco al usuario: cualquiera que conozca el password puede hacerse pasar por un usuario legitimo. Otros sistemas de autentificacion, como los biometricos, si que son (mas o menos) intrinsecos, y por ello muy dificiles de falsificar, pero su implementacion es incomoda y dificil.

El mantener un password secreto es todo un problema. Por un lado, algunos usuarios emplean passwords predecibles. Otros estan dispuestos a comunicarlo si se les persuade adecuadamente. Aveces es posible interceptar la linea por la que se comunica un password. Todos estos problemas son, o bien cuestiones de ingenieria social, o bien de

protocolos de comunicacion, y no son los que nos preocupan en este articulo. El problema con las amenazas que hemos citado esta en una misma posicion de la linea de comunicacion: el usuario.

Pero el otro extremo de la linea tambien es vulnerable: el ordenador no solo conoce el password de un usuario, si no el de todos ellos. Incluso un usuario ilegitimo con acceso parcial a la maquina podria, en teoria, sonsacar al aparato el password de un usuario con un nivel de acceso mayor, o total. Esta leccion se aprendio pronto en los sistemas en los que se consideraba que era suficiente con almacenar todos los passwords en un fichero de la maquina; un intruso avispado podria acceder a ese fichero y hacerse con el control total del sistema. No importa lo bien que se esconda el fichero, o los privilegios que sean necesarios para acceder al mismo: cualquier bug bien explotado podria dejar toda esa informacion en crudo al alcance del intruso.

La solucion obvia consiste en que la maquina no conozca en absoluto el password de los usuarios.

Esto puede parecer contradictorio. A fin de cuentas, si la maquina no conoce los passwords, no es posible que pueda autentificar a los usuarios. Sin embargo, es posible, y la idea es la siguiente: Un password es informacion, y la informacion puede transformarse de manera (practicamente) irreversible [exactamente igual que lo que sucede con la materia y la energia]. El password, al que podemos considerar como una ’causa’, se transforma irreversiblemente usando varios algoritmos, hasta producir un ’efecto’ en la forma de una cierta cadena de bytes. La realidad nos demuestra cuan dificil es deducir una causa a partir de un efecto (basta leer una novela de Agatha Christie: a partir de las evidencias, ?Quien es el asesino? :) Lo mismo se aplica a la informacion: suponiendo que los algoritmos de transformacion son buenos, es enormemente costoso obtener el password a partir del hash.

Esta es la solucion del problema. En la maquina nos limitamos a almacenar los hashes de los passwords, de manera que aunque el intruso se haga con el fichero de hashes, aun tenga una enorme barrera computacional que superar antes de meterle mano al sistema. Sin embargo, el usuario legitimo no tiene mas que comunicar el password a la maquina, la cual calcula su hash y lo compara con su base de datos. Si coincide, el usuario esta dentro.

Es importante darse cuenta de que la palabra magica aqui es ’irreversibilidad’. Esto quiere decir que no debemos usar para hashear algoritmos que sean reversibles. Por ejemplo, digamos que un implementador decide emplear el siguiente metodo: cuando la maquina pide un password en ascii, el password es truncado a 8 caracteres y luego se encripta usando DES con una clave determinada, por ejemplo ’secreto’. Este mecanismo no es irreversible: cualquiera que conozca la clave puede invertir el mecanismo en un nanosegundo. Se puede objetar que para eso antes hay que conocer la clave, pero en realidad lo que estamos haciendo es esconder un password detras de otro password: este mecanismo de hashing es seguro en tanto que la palabra ’secreto’ no sea de dominio publico. Una vez que se conozca esta palabra, todo el mecanismo se vuelve inutil.

Sin embargo, existen mecanismos de hashing de dominio publico, cuyos detalles son conocidos por todos, y que sin embargo no han sido invertidos con exito (que se sepa). Ni siquiera los que idearon esos hashes saben como invertirlos.

Naturalmente, algunos hashes son mas dificiles de invertir que otros. Con una maquina suficientemente potente y una cantidad de tiempo ilimitada, se podria invertir cualquier hash a base de fuerza bruta. La cuestion es que la barrera computacional sea lo bastante fuerte como para asegurar una seguridad temporal, idealmente mas que el tiempo que se tarda en cambiar unos passwords por otros nuevos.

Por lo tanto, el problema de los hashes se reduce a una carrera. Cuanto mas fuerte es una funcion de hashing, mas fuerte es la barrera computacional, y por tanto mas seguro esta el sistema. Se supone que las funciones de hashing se idean para no poder ser rotas en milenios, pero esto no siempre es cierto.

---| 1.2 Vulnerabilidades.

Como ya hemos dicho, toda funcion de hashing es vulnerable, dados el tiempo y el equipo suficientes. La idea es que el tiempo y el equipo sean inalcanzables en la practica. En lo sucesivo, si decimos que una funcion de hash es segura, querremos decir que es segura "en la practica".

1.2.1 Busquedas exhaustivas.

El ataque que siempre funciona consiste en hacer una busqueda exhaustiva: se recorren todos los posibles passwords, se hashean, y se comparan con el fichero de passwords. Si encontramos una coincidencia, hemos terminado. Aunque este metodo siempre tiene exito, en la practica suele ser inutil. Basta hacer algunas cuentas.

Un refinamiento consiste en hacer ataques de diccionario. En vez de buscar entre todos los passwords posibles, se emplea una coleccion plausible. Si los usuarios no tienen una buena politica de eleccion de passwords, el exito es muy probable incluso con diccionarios no muy grandes.

Un refinamiento aun mayor es hacer ataques hibridos: no solo se usa un diccionario, si no que el programa de ataque experimenta con variaciones sobre las palabras del diccionario: permutando letras, cambiando de mayusculas a minusculas, poniendo numeros o caracteres especiales al final de cada palabra, etc.

La defensa fundamental contra estos ataques es doble:

--> Primero: hay que aumentar el espacio de passwords en la medida
--> de lo posible. Es decir, que el algoritmo de hashing debe poder
--> admitir passwords de una longitud muy grande. La complejidad de
--> la busqueda crece exponencialmente con la longitud de las posi-
--> bles entradas.

--> Segundo: El algoritmo de hashing debe ser lento. Cuanto mas
--> tiempo se necesite para hacer un hash, mas tiempo se necesitara
--> para completar el ataque. Si el calculo del hash requiere un
--> segundo completo en una maquina rapida, esto es solo una ligera
--> incomodidad para el usuario legitimo, que solo tiene que iden-
--> tificarse una vez, pero para el que tiene que recorrer un
--> diccionario de miles palabras es un gran obstaculo.

Historicamente, el crypt de UNIX era un algoritmo lento, dado que se basaba en encriptar una cadena fija 25 veces usando DES. Su espacio de claves era algo reducido (8 caracteres ascii estandar), pero tampoco estaba mal para la epoca. El crypt de BSD se basa en aplicar el veloz algoritmo md5 mas de un millar de veces, lo cual lo hace mucho mas lento que el crypt clasico; su espacio de claves es ilimitado.

LM-hash se puede reducir a una unica encriptacion con DES, lo que lo hace mucho mas rapido que el viejo crypt. Su espacio de claves puede reducirse a 8 caracteres ascii (e incluso menos). NT-hash se basa en un unico hasheo con md4 (un algoritmo mas simple que md5), lo que lo hace mucho mas rapido que el viejo crypt; su espacio de claves es teoricamente ilimitado, pero usualmente se ve reducido a 14 caracteres (ya sea por causa de la GUI o por motivos de compatibilidad).

1.2.2 Precomputacion.

Suponiendo que conocemos la funcion de hashing (lo cual casi siempre es cierto), es posible precomputar enormes tablas de hashes de manera que, una vez que una tabla ha sido calculada, solo tenemos que repasarla en busca de una coincidencia con el fichero de hashes que queremos crackear. Este mecanismo es lo que se conoce como un intercambio de espacio por tiempo (’time-memory tradeoff’).

La ventaja de este metodo es que se puede hacer el calculo de antemano, y aplicarlo cuantas veces sea necesario una vez que esta hecho. La desventaja es que el calculo debe ser hecho al menos una vez, y que la necesidad de espacio para almacenamiento puede ser enorme.

Curiosamente, pueden idearse mecanismos probabilisticos de compresion muy eficientes para resolver el problema del espacio. Con el metodo adecuado, se han logrados tasas de compresion de 1000 a 1 para ciertas funciones de hashing, lo que hace que el metodo de precomputacion sea preferible al de busqueda exhaustiva en esos casos.

Supongamos que alguien desarrolle una comoda tabla comprimida que pueda romper el 99.9% de los passwords posibles para una funcion de hashing. Eso vendria a ser como un jaque mate a ese mecanismo de hashing. Sobre este problema y el LM-hash hablaremos mas adelante.

La defensa especifica contra este ataque es el ’salting’:

--> El ’salting’ consiste en definir una serie amplia de variaciones
--> sobre un mismo mecanismo de hashing. Cada variacion esta indicada
--> por una constante (conocida como ’salt’), que se almacena junto
--> al hash, y es tomada al azar cuando el hash se genera por pri-
--> mera vez. La existencia de constantes de salting multiplica la
--> talla de un diccionario eficiente por el numero de salts posi-
--> bles.

El caso clasico de salting es el crypt de UNIX. Este mecanismo de hashing admite hasta 4096 salts diferentes, lo que convierte los ataques de diccionario en practicamente imposibles. El crypt de BSD admite salts de longitud arbitraria.

Ni NT-hash ni LM-hash emplean salting.

1.2.3 Criptoanalisis.

La posibilidad de un ataque criptoanalitico es la pesadilla de todo creador de hashes. Este tipo de ataques son por lo general especificos para cada hash (o para cada familia de hashes), y suelen ser extremadamente dificiles de idear.

Los ataques sobre algoritmos criptograficos suelen mantenerse confidenciales, o en el mejor de los casos se restringen a la literatura especializada.

Aunque parezca raro, la creacion de funciones de hashing es una disciplina bastante heuristica. Aunque hay ciertas reglas bien definidas, no se puede demostrar la seguridad teorica de una funcion de hashing tal y como se demuestra un teorema de geometria. La seguridad de los algoritmos se suele determinar por su resistencia a ataques existentes, por una bateria extensiva de pruebas practicas, por algunos conceptos teoricos convincentes y, una vez que el hash ha entrado en funcionamiento, por su resistencia a los ataques que van apareciendo.

Asi que la regla de oro para confiar en una funcion de hashing concreta se basa en que no existan ataques parciales o ideas teoricas que hagan dudar de la integridad de todo el esquema.

--> Los ataques criptoanaliticos suelen venir precedidos por ataques
--> parciales contra la funcion de hashing, o contra partes de la
--> misma.

Como veremos, es mucho lo que hay que decir a este respecto en lo relativo al NT-hash y md4.

1.2.4 Malas ideas.

Malas ideas que al principio parecen fantasticas. O simplemente incompetencia, deliberada o no. Este es un error comun, y a veces tremendo.

Muchos hashes se basan en algoritmos muy fuertes de proposito general, o en conceptos teoricos avanzados, pero una implementacion inadecuada echa a perder todas las ventajas.

Un ejemplo real: cierto "super cifrado" llego a ser muy popular en una web nacional para programadores. El autor habia usado logaritmos discretos y otros conceptos avanzados para idearlo, pero debido a un error de concepto, el algoritmo se reducia a un XOR con mascara periodica. Ni que decir tiene que el autor no era un genio, pero esto le ha pasado a gente mas seria. Por ejemplo, un error en la tipificacion de LM-hash reduce drasticamente el espacio de claves (la idea de que toda letra debe pasarse a mayusculas). Que ventaja vieron los creadores de LM-hash en las mayusculas se desconoce; las ventajas que han visto los atacantes son obvias.

--> La defensa contra las malas ideas es tener una minima educacion
--> en la materia antes de ponerse a crear algoritmos. Y algo de
--> sentido comun: hay que someter todo algoritmo a una serie de
--> pruebas y a la revision de terceras personas. Esto era hace
--> tiempo mas dificil de lo que se podria creer; actualmente hay
--> mucha gente con buenos conocimentos de criptologia.

Sobre las malas ideas en LM-hash tendremos ocasion de regodearnos a continuacion. Algo podra decirse tambien de NT-hash.

----------| Parte 2 : LM-hash |----------

--| 2.1 Actualidad de LM-hash.

Si bien los desarrolladores de Windows han desechado el LM-hash en favor de NT-hash, la realidad es que por motivos de compatibilidad el LM-hash sigue siendo usado en casi todas las maquinas Windows conectadas en red. En pocas palabras, los usuarios tienen que usarlo, pero bajo su propia responsabilidad.

Es comun leer que Microsoft se defiende de haber adoptado el LM-hash alegando que lo heredo de IBM, o bien alegando que en realidad ya no es estandar, pero ninguno de estos argumentos vale para descargar la responsabilidad por los millares de servidores crackeados por idiotas usando programas de recuperacion de passwords. Y los que quedan por caer.

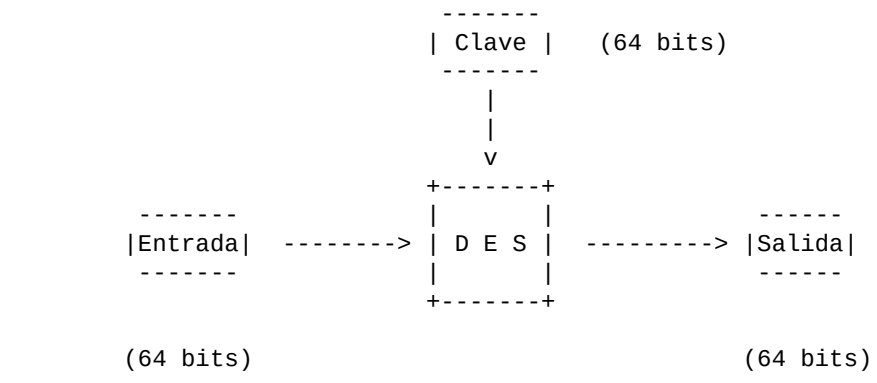
Como explicaremos en las secciones siguientes, estamos asistiendo casi con seguridad a la desaparicion definitiva de LM-hash como algoritmo de autentificacion de usuarios. Los ultimos ataques creados contra esta funcion de hashing son practicamente definitivos.

Una pregunta interesante es si los desarrolladores de Windows asumiran este hecho. Desde el punto de vista de la criptologia aplicada a la politica corporativa, es una cuestion fascinante.

--| 2.2 Implementacion de LM-hash.

2.2.1. Algo acerca de DES.

LM-hash es una funcion de hashing basada en DES. Una descripcion ligera de DES puede encontrarse en SET 20, phile 0x0d. El articulo incluye una implementacion en C, pero en la red pueden encontrarse implementaciones mucho mas potentes. En breve, DES es un cifrado simetrico que admite una clave de 64 bits (con paridad, ver mas abajo) y cifra datos en bloques de 64 bits. Esquematicamente:



Este esquema lo indicareamos abreviadamente como:

Salida = DES (Entrada , Clave)

Como DES es un criptoalgoritmo simetrico, es posible obtener 'Entrada' a partir de 'Salida' si se conoce 'Clave'. Esto lo inidicamos como:

Entrada = INV_DES (Salida, Clave)

Esto es valido para cualesquiera Entrada, Salida y Clave. Es decir, que para cada Clave de 64 bits, las transformaciones DES y INV_DES son inversas.

Sobre las propiedades mas interesantes de DES, asi como una tipificacion rigurosa del mismo, consulta [1], 7.4.2 y 7.4.3.

Salvo por el problema de la paridad, que explicaremos a continuacion, el uso de DES no suele plantear ningun problema.

Una cuestion que no suele quedar clara acerca de DES es el problema de la paridad. Aunque la clave es nominalmente de 64 bits, en realidad solo se emplean 56 de esos 64. 56 bits son 7 bytes. Esto no quiere decir que de la clave sobre un byte al final, si no que la clave es de 8 bytes, pero un bit de cada uno de esos bytes se emplea como bit de paridad (de esa manera, 1 bit * 8 bytes = 8 bits, por tanto hay 8 bits de paridad, lo que deja 7 bytes efectivos de clave). Veamos esto con detenimiento antes de continuar, porque es importante si pretendes trabajar con DES.

En primer lugar, aunque DES especifica claramente donde deben ir los bits de paridad en la clave (en el bit menos significativo de cada byte), las implementaciones de DES situan esos bits, que a fin de cuentas no se usan, donde mas les conviene. Observa por ejemplo la diferencia en la colocacion de estos bits entre la implementacion de Jhon The Ripper y la de OpenSSL.

Esto es todo un problema, por el siguiente motivo: supongamos que seguimos la implementacion de DES con rigor. Como tenemos 56 bits de clave efectiva, todo lo que podemos usar como clave son 7 caracteres ascii de 8 bits. Supongamos que nuestros caracteres son 'WELCOME' (todo un clasico para usarlo de ejemplo). En hexadecimal, 'WELCOME' es:

0x57 0x45 0x4C 0x43 0x4f 0x4d 0x45

Ahora tenemos que reagrupar estos 7 bytes en 8 grupos de 7 bits, para poder meter al final el bit de paridad. Tomando los grupos de 7 bits, y poniendolos en hexadecimal, tenemos:

0x56 0xa2 0x52 0x88 0x34 0x7a 0x34 0x8a

Finalmente tenemos que aplicar el bit de paridad a cada uno de los grupos de 7 bits (observa que todos los numeros de la linea de arriba tienen su bit menos significativo a cero). Haciendolo obtene-

mos:

```
0x56  0xa3  0x53  0x88  0x35  0x7b  0x35  0x8b
```

Esto es una clave correcta de 64 bits para DES. Si observas la descripcion de la gestion de la clave en DES (por ejemplo en [1], tabla 7.4) veras que las tablas de seleccion de bits PC1 y PC2 ignoran los bits 8,16,24, ... ,64 de la clave. En esencia, una clave para DES tiene 64 bits y paridad, pero los unicos que se emplean son 56 de esos 64. Los bits de paridad se ignoran por completo, y son tan solo una cuestion protocolaria.

La cadena 0x56, 0xa3, 0x53, ... , 0x8b es la version correcta de la clave para la implementacion de OpenSSL, pero no vale para el Jhon The Ripper, que por conveniencia coloca los bits de paridad en la parte mas significativa. Este es el motivo por el que debes tener cuidado con la paridad si trabajas con versiones prestadas de DES.

Es sencillo crear una diminuta subrutina en C que pase 7 caracteres a una clave de 8 bytes para DES. Lo siguiente puede valer:

```
typedef unsigned char des_cblock[8];

/*
 *  Convierte 7 ASCII en clave de 8 bytes.
 */

void clave7a8(des_cblock *clave)
{

    (*clave)[7] = ((*clave)[6] << 1);
    (*clave)[6] = ((*clave)[5] << 2) | ((*clave)[6] >> 6);
    (*clave)[5] = ((*clave)[4] << 3) | ((*clave)[5] >> 5);
    (*clave)[4] = ((*clave)[3] << 4) | ((*clave)[4] >> 4);
    (*clave)[3] = ((*clave)[2] << 5) | ((*clave)[3] >> 3);
    (*clave)[2] = ((*clave)[1] << 6) | ((*clave)[2] >> 2);
    (*clave)[1] = ((*clave)[0] << 7) | ((*clave)[1] >> 1);
}
```

En este codigo suponemos que los 7 caracteres estan en las posiciones (*clave)[1-7] y que la clave de 8 bytes se almacena en (*clave)[0-7] tras la conversion.

Esta subrutina no calcula los bits de paridad pero, como estos bits no son empleados por DES, no tenemos por que activarlos.

2.2.2. LM-hash paso a paso.

Vamos a dar una tipificacion de como calcular el LM-hash por etapas. Para calcular la paridad en las claves DES seguiremos el estandar DES, que usan todas las implementaciones normales (OpenSSL, CDES, etc.)

Hay un articulo de Mudge, que viene con el L0phtCrack, donde se

describe por encima lo que vamos a hacer aqui. El articulo de Mudge es menos detallado que nuestra descripcion, pero puede servirte para captar los detalles esenciales. Por otro lado, ese articulo se ocupa tambien del uso de LM-hash en protocolos de red, lo que es interesante si te atrae ese extremo del proceso de autentificacion.

Para el usuario, la implementacion de LM-hash consiste en escribir un password de hasta 14 caracteres. Si se escriben mas de 14 caracteres, el password se trunca a 14. Si se introducen menos de 14 caracteres, el password se completa con NULLs hasta llegar a los 14 caracteres. Entonces, el password se divide en dos partes, cada una de 7 bytes, que se emplean para crear sendas claves de DES. Estas claves se usan para cifrar un bloque de 64 bits fijo, 0x4b47532140232425. Esto da lugar a dos bloques (uno por cada clave). Estos dos bloques de 64 bits se concatenan en un solo bloque de 128 bits, que es el LM-hash del password introducido. Es este bloque de 128 bits el que nos encontramos, en hexadecimal, en los ficheros de passwords de Windows.

Veamos en detalle como se hace esto. Supongamos que el password que introduce el usuario es 'welcome', de menos de 14 caracteres.

Paso 1: Toma el password y pasalo a mayusculas. Se obtiene 'WELCOME'.

Paso 2: Como el password tiene menos de 14 caracteres, completalo con NULLs hasta obtener un total de 14. Se obtiene {'W','E','L','C','O','M','E',0,0,0,0,0,0,0}

Paso 3: Rompe la clave en dos partes de 7 bytes cada una. Se obtienen dos partes, que llamaremos P1 y P2:

```
{'W','E','L','C','O','M','E'}   ==  P1
{0,0,0,0,0,0,0}                 ==  P2
```

Paso 4: Expande cada parte de 7 bytes a 8 bytes de la siguiente manera: con los 7 bytes forma 8 grupos de 7 bits, y completa cada uno de esos grupos con un bit de paridad en la parte menos significativa. Esto da lugar a 8 bytes. Se obtienen dos claves, que llamaremos SK1 y SK2:

```
SK1 == {0x56 , 0xa3 , 0x53 , 0x88 , 0x35 , 0x7b , 0x35 , 0x8b }
SK2 == {  0  ,   0  ,   0  ,   0  ,   0  ,   0  ,   0  ,   0  }
```

Paso 5: Encripta con DES el bloque de 64 bits
LM == {0x4b, 0x47, 0x53, 0x21, 0x40, 0x23, 0x24, 0x25}
usando cada bloque de 8 bytes generado en 3) como clave.

```
DES(LM,SK1) ==  0xc23413a8a1e7665f
DES(LM,SK2) ==  0xaad3b435b51404ee
```

Paso 6: Concatena los dos bloques de 8 bytes obtenidos en 4) para obtener el LM-hash.

```
LM-hash("WELCOME") == 0xc23413a8a1e7665faad3b435b51404ee
```

A partir de este ejemplo es muy facil hacer una implementacion de LM-hash bastante rapida. De todas maneras, por problemas de espacio y compatibilidad (no he conseguido hacer que mi version UNIX se compile

en Windows, y viceversa), dejaremos la programacion de implementaciones concretas para el NT-hash.

Nota: Si no tienes mucha experencia con cifrados como DES, es posible que te estes preguntando como es posible que, dado que conocemos la constante LM = 0x4b47532140232425 que se usa para el cifrado, no podamos hacer mediante INV_DES una desencriptacion instantanea de las claves empleadas. El problema es sencillo. Si tenemos un LM-hash, podemos plantear las ecuaciones:

```
DES (LM, SK1) = A
DES (LM, SK2) = B
```

donde conocemos LM, A y B, y las incognitas son SK1 y SK2. Si recuerdas lo que dijimos acerca de INV_DES en 2.2.1, para poder invertir DES hay que conocer la clave, que es precisamente la incognita en este caso. Precisamente por este motivo el par de ecuaciones anterior no es trivialmente resoluble.

--| 2.3 Vulnerabilidades comunes.

Un vistazo detenido a la seccion 2.2.2 permite darse cuenta de algunos fallos evidentes del LM-hash, que se han venido empleando con mucho exito desde mediados de los 90.

a) El ataque sobre LM-hash puede dividirse en dos ataques, cada uno sobre la encriptacion DES de un bloque conocido. Si uno de estos ataques tiene exito, puede ofrecer informacion sobre la otra mitad del password.

b) LM-hash es esencialmente DES. Cualquier implementacion ultrarrapida (en software o hardware, hay de ambas) que sirva contra DES, vale directamente contra LM-hash.

c) LM-hash es un hash _muy_ rapido: una buena implementacion en ensamblador en un microprocesador moderno consume solo algunos cientos de ciclos, lo que se traduce en millones de pruebas por segundo.

d) Es muy sencillo darse cuenta de cuando un password tiene menos de 8 caracteres, dado que la segunda mitad del LM-hash sera siempre 0xaad3b435b51404ee. Esto facilita los ataques y permite seleccionar passwords ’blandos’ de un fichero simplemente echando un vistazo.

e) LM-hash no emplea salting, lo que permite hacer poderosos ataques con precomputacion. Ademas, la ausencia de salting permite atacar en paralelo tantos passwords como se desee.

f) Dado que el ataque a LM-hash puede dividirse en dos, y dado que cada subataque tiene que recuperar tan solo 7 caracteres, para muchos ataques el espacio de claves de LM-hash es de tan solo 7 caracteres (p. ej. en el caso de precomputacion).

g) El espacio de claves se ve aun mas reducido si se tiene en cuenta que las letras minusculas no se emplean en las claves.

Estos fallos se traducen en lo siguiente:

1) Un atacante con grandes recursos podra crackear cualquier LM-hash, ya sea recurriendo a precomputacion masiva, o empleando hardware especifico. El tiempo de crackeo podria reducirse facilmente a unas pocas horas, o minutos.

2) Un atacante tipico podra romper los passwords mas sencillos con gran facilidad. Segun los esquemas de ataque clasicos, la seguridad de LM-hash se basa en tomar passwords lo mas largos posibles y llenos de caracteres especiales. Esto implica que el uso de LM-hash es enormemente incomodo para el usuario que quiera un minimo de seguridad.

--| 2.4 Ataques avanzados con precomputacion.

2.4.1 Rainbow tables.

Tal y como se indico en 1.2.4, es posible hacer ataques con precomputacion y comprimir las tablas resultantes hasta que ocupen unos pocos gigabytes, lo suficiente como para caber en algunos dvds. No todos los hashes son vulnerables a este ataque (desde luego, no los que presenten salting), pero para que estos mecanismos funcionen con eficiencia deben darse algunas condiciones previas:

a) La funcion de hash debe ser lo bastante rapida como para hacer posible la precomputacion.

b) El espacio de claves debe ser lo mas reducido posible.

c) No debe haber salting.

El LM-hash califica con sobresaliente en la candidatura a las tres condiciones: DES es ultrarrapido, el espacio de claves es menor que 7 caracteres (la precomputacion ataca a cada mitad del hash), y no hay salting.

En la referencia [2] pueden observarse los resultados reales de un ataque con precomputacion y compresion, usando una nueva tecnica conocida como ’Rainbow tables’ para comprimir tablas ([2], Table 3): Usando una tabla de 2.3 gigabytes se logra un exito estimado del 99.9% a la hora de crackear passwords alfanumericos, con un tiempo medio de crackeo de 13.6 segundos en una estacion de trabajo. El autor de [2] ha logrado extender el ataque a passwords con caracteres especiales, a costa de mayor espacio de almacenamiento, pero con un tiempo y tasa de exito semejantes.

La tecnica de ’Rainbow tables’ no es, curiosamente, dificil de implementar. En el articulo citado se indica como lograrlo. El autor de este articulo ha logrado una implementacion un tanto cruda de este metodo, y ha comprobado que funciona (esta implementacion iba a ser publicada con el articulo, pero lo impidieron dos cosas: primero, una desafortunada serie de fallos de hardware de la que todavia no se ha recuperado, y segundo, que ya hay una implementacion disponible en internet, de eficacia semejante [usa la misma implementacion DES de Eric Young]).

De todas maneras, tratare de explicar brevemente como funciona este mecanismo, sin entrar en detalles muy teoricos. Si te interesan todos los detalles, no hay mas que leer [2].

Antes de entrar en detalles, imaginate el caos que puede producir que un 'hackercillo' empiece a vender dvds con las tablas precomputadas y un simple programa para revisarlas en busca de coincidencias.

2.4.2 Implementacion.

Veamos como podemos comprimir todo un diccionario de hashes en un factor, por ejemplo, de 500 a 1. Nos centraremos en el ataque a LM-hash.

La idea es formar 'hilos' de hashes donde cada uno se obtiene de los anteriores mediante una transformacion sencilla. La idea es la siguiente: para generar un hilo, partamos de una clave inicial (a la que llamaremos K0) que sera la primera de nuestro diccionario, por ejemplo 'WELCOME'(como ya hemos dicho, basta atacar 7 bytes cada vez). Entonces podemos hashear K0 para obtener un bloque de 64 bits, que como sambemos es 0xc23413a8a1e7665f. Esto es

```
                    +-----+
          'WELCOME' --> | DES | --> 0xc23413a8a1e7665f
                    +-----+
```

Ahora para generar el siguiente eslabon del hilo necesitamos un nuevo password para nuestro diccionario, que se deduzca de 'WELCOME'. Lo que hacemos es transformar 0xc23413a8a1e7665f en un password usando algun metodo; por ejemplo, yo use algo parecido a esto (para un ataque alfabetico):

```
#define COLUMN 5000

static unsigned char reasigna[] ="ABCDEFGHIJKLMNOPQRSTUVWXYZ";

static unsigned char vectores[COLUMN][7];

/*
 *  Calcula los vectores iniciales para la funcion de
 *  reduccion. Se emplean mascaras dadas aleatoriamente.
 */

void calcvect(void)
{
   unsigned int i,j,k;
   FILE *pool;

   if ((pool = fopen("/dev/urandom", "r")) == NULL)
   { perror("fopen") ; exit(1) ; }

   for (i = 0 ; i < COLUMN ; i++)
       for (j = 0 ; j < 7 ; j++)
           vectores[i][j] =  (unsigned char) fgetc(pool);

   fclose(pool);
}

/*
 *  Aplica la k-esima funcion de reduccion a un texto
 *  cifrado dado. Esta funcion lo convierte en una
 *  clave alfanumerica plausible.
```

```
 *
 *  Observa que el ultimo byte de la cifra no se usa.
 */

void freduc(des_cblock *cifra, int k)
{
    register int i;
    unsigned char c;

    for (i = 0 ; i < 7 ; i++)
    {
         c = (*cifra)[i];
         c ^= vectores[k][i];
         (*cifra)[i] = reasigna[c%26];
    }
}
```

La funcion calcvect genera una tabla de mascaras al iniciarse el programa, y lo que se usa para transformar un bloque de 64 bits en un password plausible es la funcion freduc. No te preocupes si no entiendes muy bien como marcha freduc, lo importante es lo siguiente: a partir de un bloque de 64 bits, obtenemos un nuevo password. El proceso es el siguiente:

```
             +-----+                            +------+
'WELCOME' -->| DES |--> 0xc23413a8a1e7665f -->|freduc|--> 'CZTLGJD'
             +-----+                            +------+
```

Ahora bien, una propiedad de los cifrados y hashes es que se comportan como generadores de numeros pseudoaleatorios. Esto quiere decir que como 'CZTLGJD' se ha obtenido de 'WELCOME' usando DES, en teoria no hay correlacion entre un password y otro. En otras palabras, usando DES y freduc podemos generar cadenas de passwords aleatorios.

Si llamamos K0 = 'WELCOME' y K1 = 'CZTLGJD', lo que hemos construido es un mecanismo a partir de DES y freduc (al que llamaremos Df) que hace:

```
                        +----+
     K0 = 'WELCOME' --> | Df | --> 'CZTLGJD' = K1
                        +----+
```

Ahora apliquemos este metodo 1000 veces seguidas. Esto nos permitira crear un 'hilo' de 1000 passwords tomados al azar a partir de 'WELCOME':

```
    K0 -> K1 -> K2 -> K3 -> K4 -> K5 -> ... -> K999 -> K1000
```

La clave del mecanismo de compresion es la siguiente: con tan solo conocer K0, podemos conocer todos los demas K[i] haciendo un simple calculo (aplicar Df una y otra vez).

Para generar nuestra tabla de busqueda exhaustiva lo que hacemos es quedarnos con los dos extremos del hilo, (K0, K1000). Estos dos simples passwords contendran toda la informacion sobre los passwords que estan en el medio.

Ahora generamos muchos pares de este tipo, cada uno representando

un hilo de 1001 elementos. Cuantos mas generemos, mas completa sera la tabla (en realidad, el calcular la longitud de los hilos, las funciones de reduccion, la longitud de las tablas y el numero de tablas es un interesante problema estadistico, ver [2], Sec. 5 y 7)

Supongamos que tenemos una buena tabla con los extremos de muchos hilos, calculados tras varios dias de trabajo. Ahora nos dan un hash cualquiera, llamemosle H, y queremos saber cual sera su password. Lo que hacemos es ver si el password de H esta en alguno de los hilos que hemos calculado, y es facil: Tomemos nuestro hilo (K0, K1000). Ahora apliquemos a H la funcion 'freduc'. Esto nos dara un cierto password, llamemosle K. Si K es igual a K1000, entonces es posible que K999 sea el password de H (la probabilidad exacta depende de como hayas construido las tablas. Por lo general es una muy buena probabilidad). Si K no es igual a K1000, entonces sigues adelante: Aplica a H la transformacion freduc, y luego aplica al resultado la transformacion Df. LLama al resultado K'. Si K' es igual a K1000, entonces es probable que K998 sea el password de H (probabilidad muy semejante a la anterior). Si no es asi, puedes seguir haciendo el mismo proceso aplicando ahora freduc, Df y Df. Y asi sucesivamente.

La comparacion de K, K', etc. se puede hacer simultaneamente con los extremos de todos los hilos que hemos calculado, de manera que para buscar en nuestra tabla solo tenemos que hacer 1000 revisiones: una para K, otra para K', y asi mil veces. Cada vez que K, K', etc. coincidan con un extremo de un hilo se produce lo que se llama una 'alarma': existe una posibilidad de que ese hilo contenga un password para H. Lo que hay que hacer entonces es desarrollar todo el hilo (lo que es posible dado que conocemos K0) y comparar para ver si la cosa funciona. Si funciona tenemos el password. En otro caso, tenemos lo que se llama una 'falsa alarma'.

Las falsas alarmas son una consecuencia de la naturaleza no univoca de las funciones de reduccion: freduc tiene que comprimir 64 bits en 7 caracteres alfabeticos en mayusculas, lo que supone que muchos hashes distintos se reducen en una misma clave. La naturaleza estadistica del mecanismo de compresion (usamos DES como generador de passwords aleatorios) hace que los hilos muy largos sean ineficientes, debido a que los hilos se mezclan unos con otros con cierta probabilidad. Para minimar la probabilidad de falsas alarmas manteniendo una alta probabilidad de exito hay que construir las tablas con cuidado.

El mecanismo que hemos descrito es el que se usa tanto en el ataque de 'rainbow tables' como en un ataque previo, el de Rivest. La mejora de las rainbow tables es que se reduce mucho la probabilidad de falsas alarmas usando funciones de reduccion distintas en cada punto de los hilos (es por eso que el codigo C de freduc mas arriba admite un indice entero, que permite generar muchas funciones de reduccion distintas).

Con esta introduccion al metodo es muy sencillo entender el articulo [2]. Si te interesa la criptologia, la lectura es obligatoria.

--| 2.5 Evaluacion de LM-hash.

Recapitulemos brevemente las vulnerabilidades de LM-hash basandonos en la lista que hicimos en las secciones 1.2.1 a 1.2.4. Valoraremos la amenaza de cada vulnerabilidad en la escala: 'irrelevante' 'impractiable', 'poco factible', 'factible', 'muy factible',

'peligroso' y 'critico'. Resumiremos los motivos de cada calificacion.

A) Busquedas exhaustivas: muy factible.

   El espacio de claves es muy reducido y la funcion de hashing es muy rapida.

B) Precomputacion: critico.

   Es posible generar, comprimir y distribuir tablas precomputadas con probabilidades muy altas de exito.

C) Criptoanalisis: impracticable.

   La base de LM-hash es DES, que ha resistido muchos ataques.

D) Malas ideas: peligroso.

   La posibilidad de dividir el ataque en dos y el paso a mayusculas comprometen enormemente a la funcion de hashing.

Ten en cuenta que estas valoraciones son personales. Pero los hechos que se han expuesto, creo, avalan bastante la evaluacion. Juzga por ti mismo.

----------| Parte 3 : NT-hash |----------

Suele pensarse que si LM-hash es el lado malo de la autentificacion de usuarios en Windows, el NT-hash es quien salva la situacion. De hecho, para un usuario medio el ataque a NT-hash es mucho mas dificil que el ataque al LM-hash. A pesar de todo, analicemos el NT-hash y veamos que es lo que un atacante con recursos suficientes podria ser capaz de lograr, o de haber logrado.

El NT-hash esta basado simplemente en traducir un password a UNICODE y hashearlo empleando el algoritmo md4 de Rivest (ver [3] para una descripcion completa de este algoritmo, incluyendo una implementacion en el dominio publico). Desde luego los autores de hashes corporativos no se estrujaron las meninges: para LM-hash simplemente tomaron DES tal cual; para NT-hash tomaron md4 tal cual. Tal vez para descargar la responsabilidad: si algo falla, simpre puedes colgarle el mochuelo al que invento el algoritmo original.

Comencemos con unas ideas acerca de md4.

--| 3.1 Algo acerca de md4.

Como puede verse en [3], md4 es un mecanismo de hashing concebido especificamente para ser muy rapido en maquinas de 32 bits. Conceptualmente consta de dos partes: una funcion de compresion que actua sobre bloques de datos (512 bits cada bloque) produciendo bloques de 128 bits, y un mecanismo para mezclar unos bloques de 128 bits con otros (mediante sumas modulo 2^32). La clave del algoritmo esta en la funcion de compresion, en la que se van mezclando los datos de manera (se supone) irreversible.

No mucho despues del lanzamiento de md4, su autor se dio cuenta de que hacia falta reforzarlo, lo que le llevo a la creacion de md5, con una funcion de compresion algo mas compleja. En palabras del autor [5] (ver tambien [6], Sec. 9):

> MD5 is slightly slower than MD4, but is more "conservative" in design. MD5 was designed because it was felt that MD4 was perhaps being adopted for use more quickly than justified by the existing critical review; because MD4 was designed to be exceptionally fast, it is "at the edge" in terms of risking successful cryptanalytic attack.

En la epoca de la publicacion de md5 todavia no se habian encontrado fallos criticos en md4. Sin embargo, las palabras de Rivest se han visto ampliamente confirmadas desde su publicacion, hace mas de una decada.

Es importante darse cuenta de que md4 fue concebido para producir ’huellas dactilares digitales’ de ficheros, de manera que la firma digital usando RSA fuera mas eficiente. Esto requiere un mecanismo de firma digital que ante todo sea rapido. Al contrario que DES, la base de LM-hash, md4 es un algoritmo en el que la velocidad es tan prioritaria como la seguridad.

--| 3.2 Seguridad de md4.

Una descripcion detallada de md4, incluyendo algunas de sus vulnerabilidades, se puede encontrar en [1], 9.49 y 9.50. Este libro es un buen punto de partida para entender los ataques que ha sufrido el algoritmo. Para entender que tipo de ataques son practicables sobre md4, es interesante dominar la seccion 9.2 de [1], en especial lo relativo a la resistencia a colisiones y preimagenes (ver [1], 9.2.2).

Se dice que se ha encontrado una colision en un algoritmo de hashing cuando se pueden elegir dos mensajes A y B tales que tienen exactamente el mismo hash.

La falta de resistencia a colisiones no se considera un defecto de cara a actuar como hash para passwords; simplemente significa que es posible falsificar mensajes firmados con ese algoritmo (para ver un ejemplo real de falsificacion con md4, consultar [6], Sec. 7, "Collisions for crooks" [colisiones para sinverguenzas]).

El hecho es que md4 es vulnerable a colisiones, tal y como se describe en [6]. Ataques semejantes contra variaciones de md4 [7] fueron los que llevaron a Rivest a crear md5. El calculo de una colision para md4, segun el metodo de Dobbertin, tiene una complejidad computacional de unas 2^20 evaluaciones de la funcion de compresion; teniendo en cuenta que esta funcion es extremadamente rapida, estamos hablando de un calculo de unos pocos segundos.

Una lectura detenida de [6] muestra como Dobbertin desmenuza paso a paso la funcion de compresion de md4, hasta obtener la colision. Queda claro que el hecho de que la compresion solo tenga tres etapas es totalmente insuficiente; de hecho, la principal diferencia entre md4 y md5 es que este ultimo incluye una cuarta etapa en la funcion de compresion.

Pero los problemas no se limitan a las colisiones. En la actualidad tambien existen ataques parciales a md4 como funcion de hasheo de passwords.

Si se estudia una version debil de la funcion de compresion de md4, con 2 etapas en vez de 3, es posible encontrar preimagenes para un hash dado. Es decir, que es posible crackear passwords de manera casi instantanea, tal y como afirma Dobbertin en [6]. Teniendo en cuenta que los primeros ataques de colisiones contra md4 comenzaron exactamente de la misma manera [7], es altamente probable que se pueda desarrollar (o se haya desarrollado ya en secreto) un ataque contra las tres etapas de md4. Para colmo, la peculiar estructura de UNICODE, como veremos mas adelante, podria contribuir a facilitar estos ataques en algunos casos.

Un estudio de la historia de los ataques sobre md4 permite dar completamente la razon a Dobbertin cuando afirma [6], Sec. 9:

```
+-------------------------------------------------+
| Where MD4 is still in use, it should be replaced! |
+-------------------------------------------------+
```

(el encuadrado es de Dobbertin).

--| 3.3 NT-hash paso a paso.

Vamos alla otra vez. Estudiemos como funciona y como se implementa md4, y esta vez vamos a hacerlo con codigo fuente eficiente y casero. md4 es un algoritmo tan simple, que hice una version en ensamblador de la funcion de compresion mientras veia los Simpsons.

NT-hash toma un password introducido por el usuario, y lo traduce a UNICODE. En el caso de los caracteres ascii, traducir a UNICODE consiste en extender cada byte a un par de bytes simplemente concatenando con un NULL. Asi, el caracter ascii 'a', (0x61) es en UNICODE 0x0061. La longitud del password es arbitraria, pero en muchos casos la GUI lo limita a 14 caracteres [4]. En lo sucesivo nos ocuparemos tan solo de passwords de hasta 14 caracteres.

14 caracteres unicode son 14*16 = 224 bits. Si recordamos lo dicho acerca de md4, este algoritmo comprime bloques de hasta 512 bits, de manera que para un password de 14 caracteres o menos lo unico que hace el NT-hash es aplicar la funcion de compresion de md4 al password. El resultado del hashing, de 128 bits, es almacenado tal cual en el fichero de passwords.

Con tan solo saber esto ya puedes hacer tu propio hasheador para NT; simplemente sigue estos dos pasos:

Paso 1: Traduce el password a UNICODE, lo que en muchos casos quiere decir: mete unos cuantos NULLs por enmedio.

Paso 2: Hashea con md4.

Como dijimos, en [3] hay una implementacion libre y muy eficiente de md4. No puede ser mas facil. Como alternativa, en [4] hay un crackeador usando diccionarios que emplea una implementacion de md4

del equipo SAMBA.

    Aqui tienes una implementacion en ensamblador para 'gas' de la funcion de compresion de md4:

```
/************************* Inicio de md4.S ***********************/


/* Funcion de compresion de md4, por Cemendil.
 * <cemendil@hotmail.com>
 * Este codigo esta en el dominio publico.
 */

#define ALGN 16

/*  t = A + f(B,C,D) + X[Ai] + y[Ni] ;
 *  (A,B,C,D) = (t<<>(Ls) ,B,C,D)
 *
 *  f(B,C,D) = (B&C)|((~B)&D)
 *
 *    7 ciclos.
 */

#define ROUND1(R1,R2,R3,R4,Ai,Ni,Ls) \
movl R2, %edi                     ;  \
movl R2, %ebp                     ;  \
andl R3, %edi                     ;  \
notl %ebp                         ;  \
addl 4*Ai(%esi), R1               ;  \
andl R4, %ebp                     ;  \
orl  %ebp, %edi                   ;  \
addl %edi, R1                     ;  \
roll $Ls, R1

/*  t = A + g(B,C,D) + X[Ai] + y[Ni] ;
 *  (A,B,C,D) = (t<<>(Ls) ,B,C,D)
 *
 *  g(B,C,D) = (B&C)|(B&D)|(C&D) = (B&(C|D))|(C&D)
 *
 *    7 ciclos.
 */

#define ROUND2(R1,R2,R3,R4,Ai,Ni,Ls) \
movl R3, %ebp                      ; \
movl R3, %edi                      ; \
orl  R4, %ebp                      ; \
andl R4, %edi                      ; \
andl R2, %ebp                      ; \
addl 4*Ai(%esi), R1                ; \
orl  %ebp, %edi                    ; \
addl $0x5a827999, R1               ; \
addl %edi, R1                      ; \
roll $Ls, R1

/*  t = A + h(B,C,D) + X[Ai] + y[Ni] ;
 *  (A,B,C,D) = (t<<>(Ls) ,B,C,D)
 *
 *  h(B,C,D) = B^C^D
 *
 *    5 ciclos.
```

```
 */

#define ROUND3(R1,R2,R3,R4,Ai,Ni,Ls) \
movl R2, %edi                       ; \
addl 4*Ai(%esi), R1                 ; \
xorl R3, %edi                       ; \
addl $0x6ed9eba1, R1                ; \
xorl R4, %edi                       ; \
addl %edi, R1                       ; \
roll $Ls, R1

.file "md4.S"

/* Datos iniciales. */

.data

.align ALGN

Lh1:
.long 0x67452301
Lh2:
.long 0xefcdab89
Lh3:
.long 0x98badcfe
Lh4:
.long 0x10325476


/*
 * Codigo: Se espera en la pila el puntero a una cadena
 *         de 16 longs con la cadena 'unicode' convertida
 *         a 'little endian'. El byte 0x80 de padding  y
 *         el calculo del tamaño deben estar ya hechos por
 *         el llamador.
 *
 *         El resto de la cadena _debe_ estar a cero.
 *
 *         7 longs == 224 bits == 14 chars unicode.
 *
 * Prototipo:
 *
 * (UNIX)   void _NThash(unsigned long *buf, unsigned long *out);
 * (WIN32)  void  NThash(unsigned long *buf, unsigned long *out);
 */

.text

.align ALGN

.globl _NThash
_NThash:

    pushal

    movl  36(%esp), %esi      /* %esi = &buf     */

    movl Lh1, %eax
    movl Lh2, %ebx
    movl Lh3, %ecx
    movl Lh4, %edx
```

```
ROUND1(%eax,%ebx,%ecx,%edx,0,0,3)
ROUND1(%edx,%eax,%ebx,%ecx,1,1,7)
ROUND1(%ecx,%edx,%eax,%ebx,2,2,11)
ROUND1(%ebx,%ecx,%edx,%eax,3,3,19)
ROUND1(%eax,%ebx,%ecx,%edx,4,4,3)
ROUND1(%edx,%eax,%ebx,%ecx,5,5,7)
ROUND1(%ecx,%edx,%eax,%ebx,6,6,11)
ROUND1(%ebx,%ecx,%edx,%eax,7,7,19)
ROUND1(%eax,%ebx,%ecx,%edx,8,8,3)
ROUND1(%edx,%eax,%ebx,%ecx,9,9,7)
ROUND1(%ecx,%edx,%eax,%ebx,10,10,11)
ROUND1(%ebx,%ecx,%edx,%eax,11,11,19)
ROUND1(%eax,%ebx,%ecx,%edx,12,12,3)
ROUND1(%edx,%eax,%ebx,%ecx,13,13,7)
ROUND1(%ecx,%edx,%eax,%ebx,14,14,11)
ROUND1(%ebx,%ecx,%edx,%eax,15,15,19)

ROUND2(%eax,%ebx,%ecx,%edx,0,0,3)
ROUND2(%edx,%eax,%ebx,%ecx,4,1,5)
ROUND2(%ecx,%edx,%eax,%ebx,8,2,9)
ROUND2(%ebx,%ecx,%edx,%eax,12,3,13)
ROUND2(%eax,%ebx,%ecx,%edx,1,4,3)
ROUND2(%edx,%eax,%ebx,%ecx,5,5,5)
ROUND2(%ecx,%edx,%eax,%ebx,9,6,9)
ROUND2(%ebx,%ecx,%edx,%eax,13,7,13)
ROUND2(%eax,%ebx,%ecx,%edx,2,8,3)
ROUND2(%edx,%eax,%ebx,%ecx,6,9,5)
ROUND2(%ecx,%edx,%eax,%ebx,10,10,9)
ROUND2(%ebx,%ecx,%edx,%eax,14,11,13)
ROUND2(%eax,%ebx,%ecx,%edx,3,12,3)
ROUND2(%edx,%eax,%ebx,%ecx,7,13,5)
ROUND2(%ecx,%edx,%eax,%ebx,11,14,9)
ROUND2(%ebx,%ecx,%edx,%eax,15,15,13)

ROUND3(%eax,%ebx,%ecx,%edx,0,0,3)
ROUND3(%edx,%eax,%ebx,%ecx,8,1,9)
ROUND3(%ecx,%edx,%eax,%ebx,4,2,11)
ROUND3(%ebx,%ecx,%edx,%eax,12,3,15)
ROUND3(%eax,%ebx,%ecx,%edx,2,4,3)
ROUND3(%edx,%eax,%ebx,%ecx,10,5,9)
ROUND3(%ecx,%edx,%eax,%ebx,6,6,11)
ROUND3(%ebx,%ecx,%edx,%eax,14,7,15)
ROUND3(%eax,%ebx,%ecx,%edx,1,8,3)
ROUND3(%edx,%eax,%ebx,%ecx,9,9,9)
ROUND3(%ecx,%edx,%eax,%ebx,5,10,11)
ROUND3(%ebx,%ecx,%edx,%eax,13,11,15)
ROUND3(%eax,%ebx,%ecx,%edx,3,12,3)
ROUND3(%edx,%eax,%ebx,%ecx,11,13,9)
ROUND3(%ecx,%edx,%eax,%ebx,7,14,11)
ROUND3(%ebx,%ecx,%edx,%eax,15,15,15)

addl Lh1, %eax
addl Lh2, %ebx

movl 40(%esp), %edi

addl Lh3, %ecx
addl Lh4, %edx

movl %eax,   (%edi)
movl %ebx,  4(%edi)
movl %ecx,  8(%edi)
```

```
    movl %edx, 12(%edi)

    popal

    ret


/************************ Fin de md4.S ****************************/
```

Un ejemplo de uso de esta subrutina es el siguiente:

```
/********************** Inicio de prueba.c ************************/

#include <stdio.h>
#include <string.h>

unsigned long clave[16] = {0x00610061,0x80,0,0,0,0,0,0,0,0,0,0, \
                           0,32,0};

void MDprint(long *sal)
{
  int i,j;

  for (i = 0; i < 4 ; i++)
      for (j = 0 ; j < 32 ; j+=8)
          printf("%02x",(sal[i]>>j)&0xff);
}


main()
{
    unsigned long sal[4];
    int i;

    for (i = 0 ; i < 1000 ; i++)
        _NThash(clave, sal);

    MDprint(sal);
    printf("\n");

    exit(0);
}

/************************ Fin de prueba.c ****************************/
```

El ejemplo es un poco crudo, dado que todos los datos se introducen masticados. La idea es la siguiente:

Queremos hashear el password ’aa’. Si traducimos la cadena ascii "aa" a UNICODE obtenemos 0x00 0x61 0x00 0x61. Si usamos un driver de md4 como el de Rivest [3], el driver detecta que nuestro micro (tipo Intel) es big-endian, y traduce la cadena a little-endian (asi lo requiere el algoritmo md4). Entonces pone un bit al final de nuestra cadena, calcula el numero de bits del mensaje (32 en total) y los mete al final del buffer de md4. El resultado de todas estas operaciones es la tabla de datos que usa prueba.c:

```
         {0x00610061,0x80,0,0,0,0,0,0,0,0,0,0,0,32,0}
```

El programa en ensamblador no hace ninguna de estas manipulaciones, es una tonteria programar algo asi en ensamblador, asi que le di los datos a mano al programa prueba.c. No es muy dificil hacer un programa en C que gestione el buffer.

Si quieres conocer bien la estructura del buffer de md4, lo mejor es leer la tipificacion del mismo en [3]. Es muy sencillo.

En fin, una vez organizados los datos, la rutina en ensamblador los comprime en el hash de 128 bits. Veamos como funciona el programa. Desde una shell de UNIX, mete en el directorio prueba.c y md4.S.

```
Thruspa@Killer# ls
prueba.c md4.S

Thruspa@Killer# gcc -O4 -o prueba prueba.c md4.S

Thruspa@Killer# ./prueba
c5663434f963be79c8fd99f535e7aad8
```

El NT-hash de 'aa' es 0xc5663434f963be79c8fd99f535e7aad8. Como curiosidad, he calculado el numero de ciclos que lleva cada hash en un Pentium IV a 2.4Ghz usando Cygwin. Cada funcion de compresion lleva 406 ciclos de reloj, lo que potencialmente supone mas de cinco millones de cracks por segundo. No esta mal. Una implementacion con mmx deberia ser incluso mas rapida; mi implementacion en mmx con dos hashes en paralelo me salio ligeramente mas lenta, sin embargo :(.

--| 3.4 Vulnerabilidades comunes.

Si se compara con LM-hash, el NT-hash es mucho mas duro de pelar para un atacante con pocos recursos. La unica diferencia esencial ente ambos hashings es que NT-hash tiene un espacio de claves mucho mas grande que LM-hash, dado que no es posible dividir el ataque a 14 caracteres en dos ataques sobre 7. Pero salvo esa diferencia, los dos hashings son totalmente identicos.

a) NT-hash es en casi todos los casos una funcion mas rapida que LM-hash. Una implementacion en ensamblador de NT-hash lleva en torno a 400 ciclos de reloj, mientras que una de LM-hash lleva cerca de 600 ciclos. Ambos hashes son por lo tanto tremendamente rapidos, lo que los califica bien para ataques de busqueda exhaustiva.

b) NT-hash carece de salting, lo que lo califica para ataques con precomputacion. Un ataque con rainbow tables es factible, si bien no es tan sencillo como en el caso de LM-hash. Por un lado, el espacio de claves es mayor; por otro, los passwords son en potencia el doble de largos, lo que duplica la talla de las tablas precomputadas.

En la practica, lo mas comun es lanzar ataques de diccionario sobre el NT-hash. El primer ataque de este tipo ampliamente difundido fue el de Nihil, expuesto en Phrack [4].

--| 3.5 Posibles ataques avanzados.

Como ya vimos en 3.2, md4 ha sido un gran logro para la criptologia, dado que es uno de los algoritmos que mas ataques ha recibido, evolucionando en algortimos seguros como sha y ripemd-160. El problema es que la seguridad de md4 quedo refutada en el camino.

El ataque perfecto contra NT-hash seria un ataque de calculo de preimagenes. Dado un hash, encontrar un password que se comprima en ese hash. Tal y como quedo indicado, versiones reducidas de md4 fueron atacadas con exito, generando metodos automaticos y rapidos para encontrar preimagenes, ya en 1997.

Segun el criterio generalmente aceptado, un ataque parcial sobre un sistema permite dudar sobre la seguridad de ese sistema (1.2.3). La derrota completa de md4 en lo relativo a colisiones y falsificacion de mensajes puede sumarse como otra advertencia seria sobre la debilidad de md4. En resumen, es altamente plausible que exista un ataque de preimagenes contra md4, seguramente clasificado como alto secreto.

El NT-hash no refuerza precisamente al algoritmo md4. Hay que tener en cuenta que el buffer de md4 en un password de NT-hash es altamente predecible: para la mayor parte de los caracteres que se encuentran en un password normal, el formato UNICODE exige la presencia de un NULL antes del numero ascii de ese caracter. Un vistazo al articulo [6] demuestra lo mucho que se puede hacer cuando se conoce una parte del buffer de md4. Pues bien, en un password tipico de menos de 14 caracteres mas de un 60% del buffer de md4 es altamente predecible.

El autor (Cemendil) solo ha trasteado un poco con lo que se podria sacar criptoanalizando seriamente esta debilidad, pero alguien con mas talento bien podria haber sacado partido de ello. La oportunidad, cuando menos, es atractiva.

--| 3.6 Evaluacion de NT-hash.

Pasemos, como ya hicimos en 2.5, a hacer un resumen de los puntos fuertes y debiles de NT-hash desde la perspectiva de la seccion 1.2.

A) Busquedas exhaustivas: poco factible.

   Aunque el espacio de claves es muy grande, los ataques de diccionario e hibridos son faciles y eficientes. El algoritmo es extremadamente rapido.

B) Precomputacion: poco factible.

   Es posible generar, comprimir y distribuir tablas precomputadas con razonables probabilidades de exito, si bien las tablas seran necesariamente mucho mayores e incomodas que en el caso de LM-hash.

C) Criptoanalisis: peligroso.

   La base de NT-hash es md4, que ha sido criptoanalizado con exito en dos escenarios distintos: colisiones (exito completo) y preimagenes (exito parcial).

D) Malas ideas: factible.

El uso de UNICODE, y la limitacion a 14 caracteres en algunos casos, causan una considerable predecibilidad en el buffer de md4.

Una vez mas, advierto que estas apreciaciones son personales, si bien creo haber expuesto los hechos con bastante precision. Lo unico que se me ocurre es repetir la frase de Dobbertin: Where md4 is still in use, it should be replaced!

----------| Conclusiones |----------

Los algoritmos de autentificacion de usuarios en los sistemas operativos Windows son inadecuados. Ambos comparten unas caracteristicas sospechosas: en el momento de su creacion eran lo bastante fuertes como para resistir el ataque de individuos, pero no eran lo bastante buenos como para resistir ataques poderosos y bien financiados. Con el tiempo, la situacion de LM-hash, el mas viejo de los dos, se deterioro hasta el punto de que algunos hackers hacian dinero vendiendo crackeadores altamente eficientes. Finalmente, los ataques por precomputacion han liquidado a LM-hash como un mecanismo de seguridad valido. La situacion de NT-hash es mas fuerte, pero solo porque su espacio de claves es mayor, de manera que no es vulnerable a los ataques de fuerza bruta que tan bien funcionan con su antecesor. Sin embargo, todo indica que tarde o temprano se ideara un metodo de ataque criptoanalitico que liquidara tambien a NT-hash.

La situacion es irritante si se compara con el estado del hashing en el mundo de UNIX. Incluso si los desarrolladores de Windows hubieran adoptado el venerable crypt de UNIX, su situacion seria mejor de lo que es en la actualidad. Todavia habria sido mejor que hubieran tomado metodos de hashing mas avanzados; los tenian a mano por todas partes. El que no los usaran permite pensar que quizas no quisieron hacerlo, sin que por eso se me tenga que calificar de paranoico.

Las maquinas basadas en Windows contienen multitud de datos sensibles y sirven de apoyo a una parte importante de la economia mundial. Seria deseable que sus sistemas de autentificacion de usuarios fueran reforzados, en vista de que son claramente insuficientes.

----------| Referencias |----------

[1] A. Menezes, P. van Oorschot, S. Vanstone,
"Handbook of Applied Cryptography"
CRC Press, 1996

hhtp://www.cacr.math.uwaterloo.ca/hac

[2] Philippe Oechslin,
"Making a Faster Cryptanalytic Time-Memory Trade-Off"
LASEC, Ecole Polytechnique Fédérale de Lausanne.

http://lasecwww.epfl.ch/

[3] R. Rivest,
"The MD4 Message Digest Algorithm"

RFC 1186, October 1990.

[4] Nihil,
"Cracking NT passwords"
Phrack Magazine, Issue 50, Phile 0x08

http://www.phrack.org

[5] R. Rivest,
"The MD5 Message Digest Algorithm"
RFC 1321, April 1992.

[6] Hans Dobbertin,
"Cryptanalysis of MD4"
J. of Cryptology (1998) 11:253-271

[7] Hans Dobbertin,
"RIPEMD with Two-Round Compress Function is not
Collision-Free"
J. of Cryptology (1997) 10:51-69

[8] Hans Dobbertin,
"The Status of MD5 After a Recent Attack"
RSA Laboratories' Crypto Bytes, Volume 2, Number 2
Summer 1996

EOF

```
-[ 0x12]--------------------------------------------------------------------
-[ Llaves PGP]---------------------------------------------------------------
-[ by SET Staff ]-----------------------------------------------------SET-29--

PGP <http://www.pgpi.com>

Para los que utilizan comunicaciones seguras, aqui teneis las claves publicas
de algunas de las personas que escriben en este vuestro ezine.

<++> keys/grrl.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP 6.0.2

mQDNAzcEBEcAAAEGANGH6CWGRbnJz2tFxdngmteie/OF6UyVQijIY0w4LN0n7RQQ
TydWEQy+sy3ry4cSsW51pS7no3YvpWnqb135QJ+M11uLCyfPoBJZCcIAIQaWu7rH
PeCHckiAGZuCdKr0yVhIog2vxxjDK7Z0kp1h+tK1sJg2DY2PrSEJbrCbn1PRqqka
CZsXITcAcJQei55GZpRX/afn5sPqMUslOID00cW2BGGSjtihplxySDYbLwerP2mH
u01FBI/frDeskMiBjQAFEbQjR2FycnVsbyEgPGdhcnJ1bG9AZXh0ZXJtaW5hdG9y
Lm5ldD6JANUDBRA3BARH36w3rJDIgY0BAb5OBf91+aeDUkxauMoBTDVwpBivrrJ/
Y7tfiCXa7neZf9IUax64E+IaJCRbjoUH4XrPLNIkTapIapo/3JQngGQjgXK+n5pC
lKr1j6Ql+oQeIfBo5ISnNympJMm4gzjnKAX5vMOTSW5bQZHUSG+K8Yi5HcXPQkeS
YQfp2G1BK88LCmkSggeYklthABoYsN/ezzzPbZ7/JtC9qPK4O7Xmjpm//ni2E10V
GSGkrcnDf/SoAVdedn5xzUhHYsiQLEEnmEijwMs=
=iEkw
-----END PGP PUBLIC KEY BLOCK-----
<-->


Tipo Bits/Clave    Fecha      Identificador
pub   768/AEF6AC95 1999/04/11 madfran <madfran@nym.alias.net>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3ia

mQBtAzcQ8VIAAAEDAJuWBxdOxP81fhTJ29fVJ0NK/63dcn5D/vO+6EY0EHGHC42i
RF9gXnPUoSrlNfnfFnF9hZO0Ndb4ihX9RLaCrul8+FN97WYCqSonu2B23PpX7U0j
uSPFFqrNg0vDrvaslQAFEbQfbWFkZnJhbiA8bWFkZnJhbkBueW0uYWxpYXMubmV0
PokAdQMFEDcQ8VPNg0vDrvaslQEBHP0C/iX/mj59UX1uJlVmOZlqS4I6C4MtAwh3
7Dh5cSHY0N0WBRzSBKZD/O7rV0amh1iKkrZ827W6ncqXtzHOsQZfo183ivHOc3vM
N4q3EEzGJb9xseqQGA61Ap8R8rO37Q8kEQ==
=vagE
-----END PGP PUBLIC KEY BLOCK-----


Tipo Bits/Clave    Fecha      Identificador
pub   768/7E6141FD 2003/02/02 The KSTOR <kstor@nym.alias.net>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3ia

mQBtAz49hkwAAAEDAPjRz2+4hxuVK5trm//nWuRNbNOgsv5Ab4m4eHXZKPhvcgv8
3gn+OGvwfXg6u/6JUMotdu1ZUXzVCQnK4N9izymRci2MHBTdD3ppnaR2F8zW5okj
dKYVfYVEjdEBfmFB/QAFEbQfVGhlIEtTVE9SIDxrc3RvckBueW0uYWxpYXMubmV0
PokAdQMFED49hkxEjdEBfmFB/QEBMFYC/iUC2fcwngqDzf3B6Rsa1Cb/vs50hnJX
ijLnghNjiLHdz162oz8pejvc8b1eRWS9cFuPKxm6aanHOk/JF8jedcT62zHkdJrl
Igzzku3qflJFz/dy1EIcAuJm/woVDDbuSA==
=qDFc
-----END PGP PUBLIC KEY BLOCK-----

  ú---------[ ULTIMA ]-------------ú--------------------------------------ú
  |                                |                                      |
  ú---[ ULTIMA NOTA ]--------------ú                                      |
```

"A partir de manyana, los empleados van a entrar al edificio usando Tarjetas individuales de seguridad. El proximo miercoles se les van a tomar fotografias y en dos semanas recibiran la tarjeta."

Fred Dales, Microsoft Corporation

SET, - Saqueadores Edicion Tecnica -. Numero #29
Saqueadores (C) 1996-2004

EOF