ALGUNOS CONCEPTOS TÉCNICOS

(pocos, breves... y necesarios)

El texto de esta sección ha sido redactado tomando como base de información la contenida en diversos artículos de Wikipedia, y se enriquece con numerosos enlaces que conducen a textos muy ilustrativos y completos. Pero *NO ES NECESARIO que usted los consulte ahora*. Hágalo sólo cuando sienta curiosidad y quiera ampliar los conocimientos que le son ofrecidos aquí: éstos son suficientes para comprender con claridad qué es el software libre.

Contenido:

---

1. Lenguaje de programación

Cuando el desarrollador de un programa encara su tarea, lo primero que hace es elegir un lenguaje de programación, elección que va a estar determinada por las funciones a cumplir por el programa proyectado.
Algo así como cuando usted, que domina varios idiomas, elige para comunicarse el que se habla en el país a donde llega.
Al igual que los lenguajes naturales, los lenguajes de programación son un conjunto de vocablos, sintaxis y reglas semánticas que definen los programas del computador.
Si quiere conocer más sobre el tema, puede hacer un paseo por los enlaces que siguen: son diversos lenguajes de programación.

Ada | Algol | Awk | Basic| BCPL | C | C++ | C# | Cobol |
Delphi | Eiffel | Forth | Fortran | Haskell | Java | Lisp | Logo |
MAGIC | Miranda | Modula-2 | Oberon | Ocaml | Pascal |
PHP | Perl | Prolog | Python | Ruby | Smalltalk |
Otros lenguajes...

Pero… si luego de dar una vuelta por los lenguajes halla que no entiende nada, no se preocupe: aquí basta con lo dicho para que usted tenga una noción, suficiente a nuestro fines, de qué es un lenguaje de programación.

2. Código fuente

El código fuente no es otra cosa que un texto escrito, simple.
Un texto similar al que usted genera cuando usa el "block de notas" de su sistema Windows; un texto plano -es decir, sin formatos- rigurosamente ceñido al vocabulario y a las reglas semánticas y de sintaxis propias del lenguaje elegido por el programador.

Aquí va un ejemplo:

```
program sumatorio
implicit none
integer, parameter :: MAX=100    !Numero maximo de elementos
integer :: num_elem, &       !Numero de elementos
i                                !Contador
real, dimension(MAX) :: v        !Vector
logical :: default

write(*, *) 'Introduzca el numero de elementos: '
read(*, *) num_elem
write(*, *) 'Introduzca uno a uno los elementos: '
read *, (v(i), i=1, num_elem)
print *, 'Suma = ', sum(v, num_elem)
print *, 'Todo listo.'
end program sumatorio

function sum(v, n)
real v(n)
sum = 0.0
do i = 1, n
sum = sum + v(1)
end do
return
end function
```

El código fuente es texto simple, capaz de ser leído por cualquier editor de textos y lo que es más importante, **comprensible por cualquier programador** que conozca el "idioma" utilizado. En él están escritas las instrucciones que deberá realizar la computadora, según la sintaxis de un lenguaje de programación.

Conocer el código fuente es
imprescindible
si se quiere estudiar o modificar
o saber cuáles son las reales funciones
de un programa de computación.

3. Compilador

El lenguaje de programación utilizado para la redacción del código fuente es lo que se denomina un lenguaje de alto nivel, esto es, un lenguaje en que el programador puede expresarse con cierta facilidad por ser el más próximo al lenguaje natural.
Pero... hay un problema:

La máquina no lo entiende.

La máquina sólo comprende el lenguaje de máquina, un "idioma" en el que los humanos no podemos expresarnos.
Este desentendimiento lo resuelve un programa-herramienta: el compilador.
Comparando su actuación con la de un ser humano, un compilador equivale a un traductor profesional que, a partir de un texto, redacta otro, independiente, plasmado en un ejemplar nuevo traducido a otra lengua.
Nuestro programador echará mano al compilador y, teniendo como base los archivos en código fuente por él desarrollados, generará otro "código": **el código objeto**.

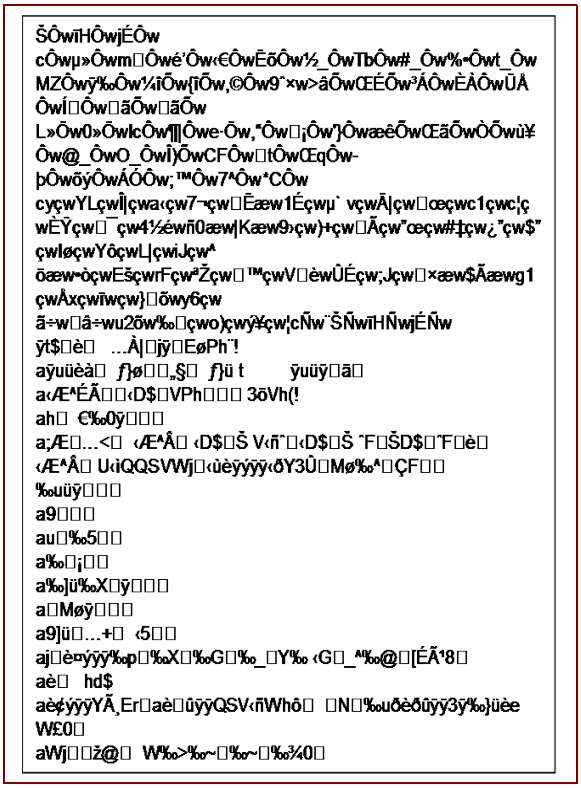
> **Nota**: en esta fase podría utilizarse otro programa-herramienta: el intérprete.
> Nuevamente: comparando su actuación con la de un ser humano, un intérprete equivale al traductor humano que de viva voz traduce las palabras que oye sin dejar constancia por escrito, tal como lo hace, por ejemplo, el intérprete en un evento académico.
> Pero ése no es el proceso que nos interesa aquí. Si usted quiere conocer algo más sobre el tema, puede comenzar pulsando la palabra "intérprete" señalada como hipervínculo al comienzo de esta nota.

4. Código objeto

Resumiendo hasta aquí: el programador escribe el código fuente con el lenguaje de programación elegido, y, en un proceso de compilación, este código fuente se convierte en código objeto.

Podemos, entonces, definir el *código objeto* como el archivo que resulta de compilar el archivo del código fuente.

Este código objeto está expresado, ahora sí, en el único "idioma" que entienden las computadoras. Las computadoras u ordenadores trabajan internamente con dos niveles de voltaje, lo que se representa simbólicamente con dos dígitos: "1", voltaje más alto; "0", voltaje más bajo. Esto no es otra cosa que el código binario que permite a la máquina ejecutar las instrucciones que el programador escribió en forma de código fuente.
Por lo dicho, el archivo código objeto se denomina también archivo binario o archivo ejecutable.
Si intentamos abrir un archivo binario, la máquina tratará de mostrarlo y lo que veremos será más o menos esto:



Ininteligible ¿verdad?

El código objeto, por estar expresado en binario, **no permite, a partir de él, conocer el código fuente.** De lo que se desprende una grave consecuencia:

Sin acceso a su código fuente,
no podemos saber qué es lo que
realmente hace un programa.

Y este *no saber qué es lo que realmente hace un programa* es lo que habilita la existencia de código subrepticio, malicioso, espía o de control remoto.

El software libre, **sometido a un proceso de revisión pública permanente**, hace posible que las "puertas traseras" introducidas en un programa sean detectadas prácticamente de inmediato; y que la comunidad informática mundial -trabajando coordinadamente vía internet- en pocas horas aporte el recurso para eliminar la anomalía.

Sólo el código fuente abierto garantiza la seguridad informática,

es lo único que satisface el que ha sido siempre objetivo prioritario de los desarrolladores en el "mundo linux": el de la seguridad, tanto del sistema cuanto de los datos, lo que implica también garantizar la privacidad del usuario.

Es frecuente que el usuario estándar se pregunte si un sistema "abierto" es seguro. Este interrogante es consecuencia de la confusión que identifica la *exposición pública de lo que hace un sistema informático* con la *exposición pública de los datos procesados por ese mismo sistema*.
El cómo funciona su computadora no es preocupación del usuario estándar, por cierto. Pero que no pueda saber de ninguna manera cuáles son las instrucciones que ejecuta su computadora porque el código fuente de los programas en ella cargados es secreto rigurosamente guardado por el licenciante del producto... bueno, eso sí debería generar su mucha preocupación.
Si el usuario de aplicaciones propietarias ha sido precavido, tendrá instalado en su equipo antivirus, antiespías, firewalls. ¿Por qué? Porque sabe que programas subrepticios pueden "colarse" vía correo electrónico, vía internet, vía intercambio de archivos. Esto, que durante un par de décadas se nos enseñó a aceptar como inherente al uso de la informática, es una grave patología que se soluciona con el uso de sistemas abiertos, en los que el diseño abierto del hardware y el código abierto del software excluyen la posibilidad de "puertas traseras" que violen su privacidad y la seguridad de sus datos.

Quien esto escribe -y apuesto a que también quien esto lee- seguramente no estamos en condiciones de leer un código fuente ni de entender los planos de un componente de hardware.
Pero si tengo en mi mano un texto en alemán -idioma que no comprendo- tengo la posibilidad de convocar a un traductor para que me informe de lo que dice. No la tendré, en absoluto, si el texto está cifrado y el creador de la clave guarda ésta para sí.

5. Sistema Operativo

Un *sistema operativo* es:
un conjunto de programas destinado a

- **permitir la comunicación del usuario con la computadora y**
- **gestionar sus recursos de hardware de manera cómoda y eficiente.**

Definición de Wikipedia

De los muchísimos programas que componen el sistema operativo, el que nos interesa en particular saber de qué se trata es el denominado **kernel**, palabra de uso frecuente en el "lenguaje linux".
El kernel, también llamado **núcleo**, es la parte fundamental de un sistema operativo. Es el responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora.
Para aclarar la idea, pensemos en un objeto que nos es familiar: el automóvil. Y comparémoslo con un sistema operativo:

AUTOMÓVIL	SISTEMA OPERATIVO
Partes de la maquinaria: caja de velocidades, radiador, carburador, transmisión, diferencial...	**Programas del SO:** *gestionan* procesos, el uso de memoria, archivos y directorios, la entrada y salida de datos, la seguridad y protección, la comunicación y sincronización entre procesos, la interpretación de órdenes...
MOTOR	**KERNEL o NÚCLEO**

Hasta aquí, el sistema operativo.

Y si utilizamos la misma comparación, pero ahora entre un automóvil y los programas de usuario o aplicaciones, podríamos diseñar lo siguiente:

AUTOMÓVIL	APLICACIONES
Elementos que maneja o utiliza el usuario: **butacas, volante, pedales, guantera, limpiaparabrisas, ventanas, levantavidrios, baúl...**	**Aplicaciones:** **procesador de textos** **planilla de cálculo** **navegador web** **mensajero instantáneo** **cliente de correo…**

Veámoslo en un ejemplo.

Cuando el conductor de un automóvil en movimiento quiere pasar de una marcha a otra, lo que hace es pisar el pedal de embrague y mover de determinada manera la palanca de cambios. No son estas acciones las que provocan por sí mismas el cambio de velocidad -de hecho, nada sucedería si el automóvil tuviera su motor apagado-, sino que operan a modo de instrucciones para que en la caja de cambios se produzca un desacople de engranajes, una sustitución en cierta posición de un piñón de determinadas características por otro diferente, y un nuevo acople, de lo cual derivará una modificación en el desempeño del motor; modificación que, a su vez, otros componentes mecánicos transmitirán desde el motor a las ruedas, en donde -finalmente- se exteriorizará el efecto del accionar del conductor.

De modo análogo, el usuario da instrucciones a su computadora a través de los periféricos de entrada -teclado, mouse, micrófono, tableta digitalizadora, etc.,-las que serán recibidas, primero por la aplicación de que se sirve -procesador de textos, planilla de cálculo, programa de diseño, etc.-, y luego por el sistema operativo, que será el encargado de transmitirlas al hardware -memoria RAM, placas de sonido y vídeo, microprocesador, etc.- para su procesamiento y devolución, y cuyo resultado llegará al usuario a través de los periféricos de salida -pantalla del monitor, impresora, parlantes, etc.-.

En ambos casos se trata de procesos "en capas", de las cuales sólo son visibles la capa donde operan las instrucciones desencadenantes y la capa en que se hace manifiesto su efecto final.
Lo que no vemos, en el automóvil está a cargo de sus partes mecánicas y del motor; y en el ordenador, a cargo del sistema operativo.

6. Distribución

Éste no es un concepto técnico, pero "distribución" es una palabra que oirá siempre al hablar de software libre. La explicamos:

Una *distribución* es:
un conjunto de aplicaciones agrupadas con el propósito de permitir instalar fácilmente un sistema GNU-Linux.

Se compone de

- **el sistema operativo en sí, *más***
- **el instalador, *más***
- **gran cantidad de aplicaciones, juegos y utilitarios.**

Aquí concluye esta introducción a conceptos técnicos ("*pocos, breves y... necesarios*") cuyo conocimiento facilitará al lector la comprensión de qué es el software libre y le servirá de base para avanzar con paso firme en su recién iniciado proceso de capacitación.