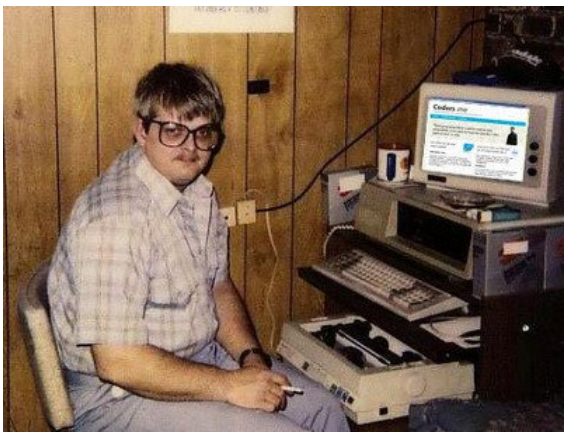
Números Mágicos, Los Códigos Que Esconden Los Programadores



En algún momento hemos oido hablar de los famosos huevos de pascuas, aquellos pequeños códigos dejados por los programadores para dejar un pequeño mensaje a los usuarios. Si alguna vez has sido curioso habrás escrito alguna palabra en tu calculadora usando números, y aquellos programadores de naturaleza nerd han estado **escondiendo números secretos** en el interior de su PC, y usarlos para algunos «apretones de manos» secretos entre las aplicaciones y los archivos.

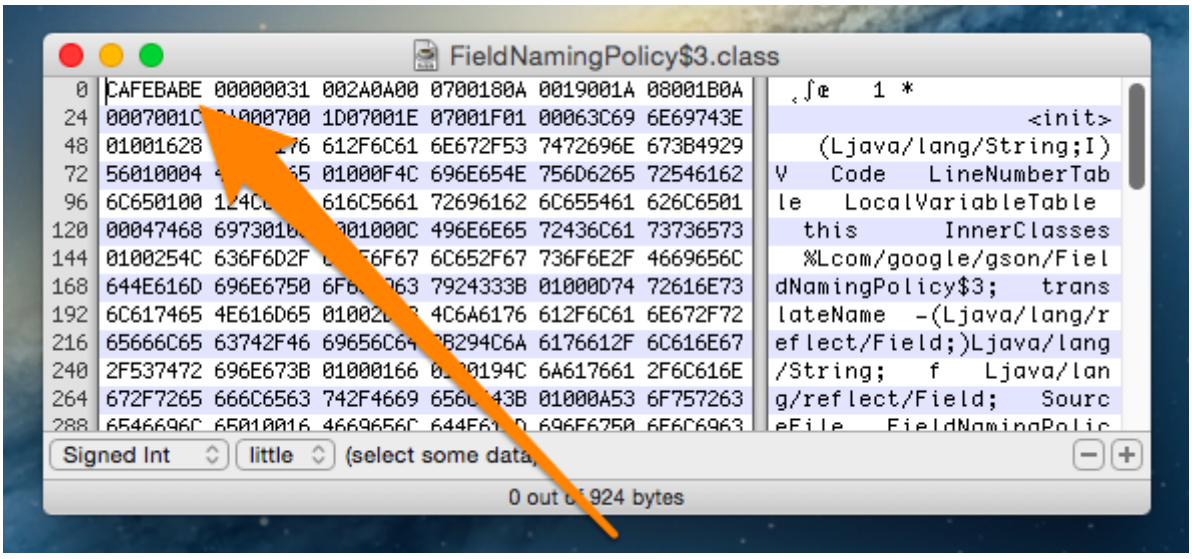
En este articulo te mostraremos algunos de los ejemplos más entretenidos que encontramos sobre los números mágicos o Magic Numbers ;)

¿Qué son los números

mágicos?

La mayoría de los lenguajes de programación usan un tipo entero de 32 bits para representar ciertos tipos de datos en background, internamente el número se almacena en la memoria RAM ó es **utilizado por la CPU** como unos y ceros, pero en el código fuente serían escritos en cualquiera de los dos formatos decimal regular, o como **formato hexadecimal**, que utiliza los números del 0 al 9 y las letras A a F.

Cuando el sistema operativo ó una aplicación quiere **determinar el tipo de un archivo**, se puede mirar hacia el principio del archivo, en un marcador especial que significa el tipo de archivo. Por ejemplo, un archivo PDF podría comenzar con el **valor hexadecimal 0x255044462D312E33**, lo que equivale a «**%PDF-1.3**» en **formato ASCII** ó un archivo ZIP que comienza con 0x504B, lo que equivale a «PK», que desciende de la **utilidad original de PKZip**. Al observar esta «**firma**«, un tipo de archivo se puede identificar fácilmente, incluso sin ningún tipo de otros metadatos.

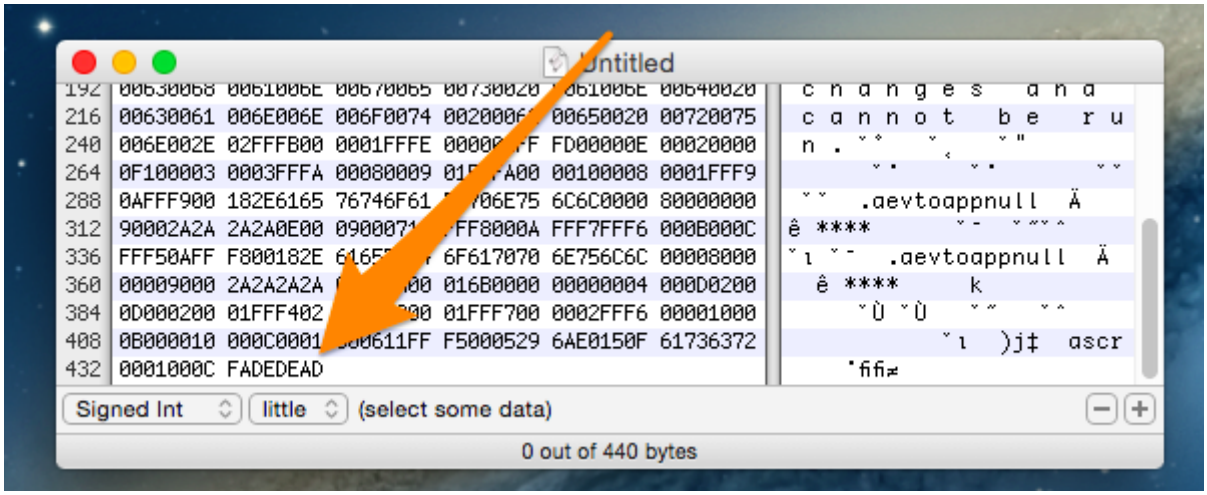


La utilidad de Linux «file» se puede usar desde el terminal para determinar el tipo de un archivo, de hecho, lee los

números mágicos de un archivo llamado «magic». Cuando una aplicación desea llamar a una función, se puede pasar valores a esa función utilizando tipos de datos estándar como número entero, que se puede expresar en el código fuente en formato hexadecimal. Esto es especialmente cierto para las constantes, que son identificadores definidos con nombres legibles como **AUTOSAVE_INTERVAL**, pero el mapa de número entero real (u otro tipo) valores. Así que en lugar de un programador escribir un valor como 60 cada vez que llama a la función en el código fuente, que podían usar la **constante AUTOSAVE_INTERVAL** para una mejor legibilidad. (Las constantes son generalmente fáciles de reconocer porque están escritas en letras mayúsculas).

Algunos ejemplos notables de los Magic

Numbers



Si das una mirada rápida en el código fuente de Linux, verás que la llamada a _reboot () requiere una variable «mágica» para ser pasado que es igual al número hexadecimal **0xfee1dead**. Si hay algo que trató de llamar a esa función sin pasar en ese valor mágico, sólo devolverá un error.

El **GUID** (identificador único global) para una partición de arranque de la BIOS en el esquema de particiones GPT es 21686148-6449-6E6F-744E-656564454649, que forma la cadena ASCII «**Hah! IdontNeedEFI**«, una alusión al hecho de que normalmente se utilizaría GPT en equipos que sustituyó BIOS con UEFI, pero no necesariamente tiene que ser así ;)

El famoso Microsoft escondió **0x0B00B135** en su código fuente de soporte para una máquina virtual de Hyper-V presentado a Linux, entonces cambiaron el valor a 0xB16B00B5, y finalmente cambiaron a decimal antes de ser retirado a partir del código fuente completo.

Los ejemplos más divertidas incluyen:

- 0xbaaaaaad — utilizado por el registro de accidente de iOS para indicar que un registro es un stackshot de todo

el sistema.

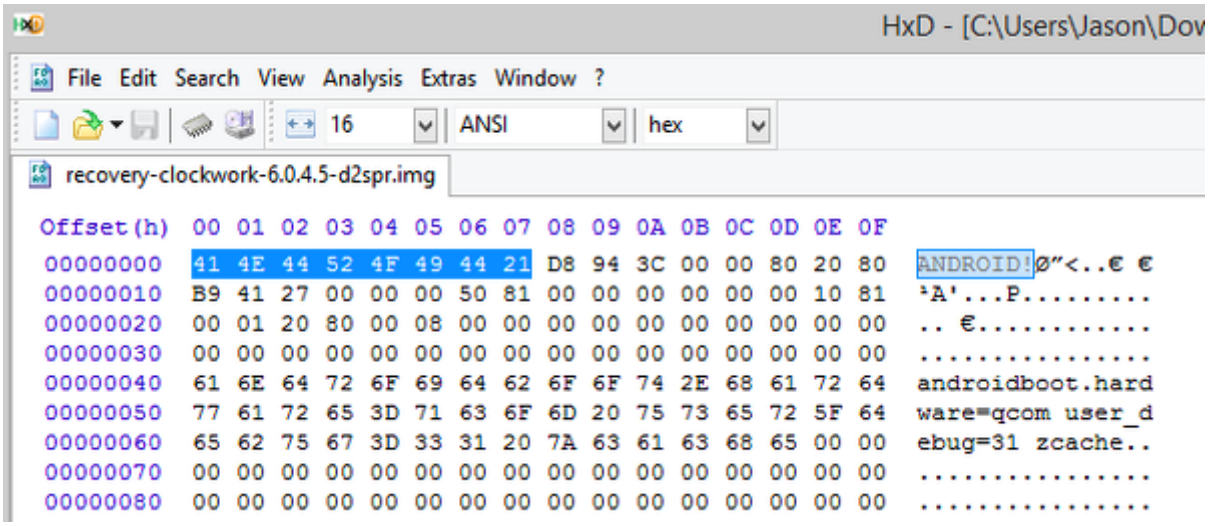
- 0xbad22222 – utilizado por el registro de accidente de iOS para indicar que una aplicación de VoIP ha sido asesinada por iOS porque funcionaba mal.
- 0x8badf00d – (comiendo mal) utilizado por los registros de accidentes de iOS para indicar que una aplicación tardó demasiado en hacer algo y fue asesinado por el vigilantes de tiempo de espera.
- 0xdeadfa11 – (Caída Muerto) utilizado por iOS tala accidente cuando una aplicación es forzar la salida por un usuario.
- 0xDEADD00D – utilizado por Android para indicar un aborto VM.
- 0xDEAD10CC (Dead Lock) utilizado por iOS cuando una aplicación se bloquea por un recurso en background.
- 0xBAADF00D (mala comida) utilizado por la función LocalAlloc en Windows para la depuración.
- 0xCAFED00D (Cafe tio) utilizado por compresión pack200 de Java.
- 0xCAFEBABE (Cafe nena) utilizada por Java como el identificador de archivos de clase compilados
- 0x0D15EA5E (Enfermedad) utilizado por Nintendo en la Gamecube y Wii para indicar un inicio normal pasó.
- 0x1BADB002 (1 mal arranque) utilizado por la especificación de arranque múltiple como un número mágico
- 0xDEADDEAD – utilizado por Windows para indicar un accidente de depuración iniciado de forma manual, también conocida como la pantalla azul de la muerte.

Por supuesto, estos no son los únicos que hay, pero sólo es una breve lista de ejemplos. Conoces algunos otros? Cuéntanos en los comentarios.

Al ver ejemplos para usted mismo

Puede ver más ejemplos mediante la apertura de un editor hexadecimal y luego abrir cualquier tipo de archivo. Existe un montón de **editores hexadecimales** gratuitos disponible para Windows, OS X o Linux. Sólo asegúrese de tener cuidado al instalar software gratuito para no infectarse con crapware o spyware.

Como un ejemplo adicional, las imágenes de recuperación para los teléfonos Android como ClockworkMod comienzan con «ANDROID!» Si se lee en formato ASCII.



Nota: No modifiques nada mientras que estás mirando el archivo. Los editores hexadecimales pueden romper cosas!