Marcos Loiseau

"La codicia y la locura doble del sufrimiento en la suerte del hombre."
-Homer

"Hola a todos" en 32 bits Asamblea Linux (NASM)

Publicado el **30 de abril 2012**

Escribir un programa de 32 bits "Hello World" en NASM es un buen primer paso para cualquier persona que quiera aprender el montaje de Linux. Tanto si eres un programador que quiera probar alguna optimización de montaje o un shellcoder aspirante, no hay evitando "Hello World" (a menos que la suerte y encontrar un conjunto quine tutorial). Guías de montaje de Linux en profundidad con los antecedentes completos están disponibles en todo el lugar, comenzando con manual de NASM de Sourceforge .

La rutina de "Hello World" a continuación podría servir como una buena guía de inicio rápido para Linux NASM, o el montaje de Linux en general. Se incluye el código fuente NASM y las instrucciones de compilación de breves.

¿Cómo escribir, compilar y ejecutar un programa "Hola Mundo" en 32 bits asamblea Linux (NASM):

1. Instale NASM.
2. Escriba o copie / pegue el archivo de origen del ensamblado
3. Compile el ensamblado en código de máquina con NASM.
4. Vincular el código compilado con gcc o ld.
5. Ejecute el ejecutable ELF resultante.

Update: Usted puede descargar el código fuente comentado y un Makefile conveniente para los NASM programas de 32 bits y 64 bits "Hello World" aquí (tarball). Todo está automatizado.

1. Instale NASM y build-essential

Solía apt-get para instalar NASM en mi equipo. Es importante tener en cuenta que lo hice en una versión de 64 bits de Ubuntu 11.10. Las llamadas al sistema y las herramientas pueden ser muy diferentes entre distribuciones. Este proceso puede no funcionar en absoluto en otras versiones o

distribuciones. No puedo responder.

```
$ sudo apt-get install nasm build-essential
```

2. Cree el origen de ensamblaje

Descarga el código fuente o copiar y pegar desde aquí:

```
; "Hello World!" in 32 bit Linux NASM
; adapted from http://asm.sourceforge.net/intro/hello.html by Mark Loiseau
; referenced in http://blog.markloiseau.com/2012/04/hello-world-NASM-Linux

global _start            ; global entry point export for ld

section .text
_start:

    ; sys_write(stdout, message, length)

    mov eax, 4         ; sys_write syscall
    mov ebx, 1         ; stdout
    mov ecx, message     ; message address
    mov edx, length        ; message string length
    int 80h

    ; sys_exit(return_code)

    mov eax, 1         ; sys_exit syscall
    mov ebx, 0         ; return 0 (success)
    int 80h

section .data
    message: db 'Hello, world!',0x0A     ; message and newline
    length: equ $-message            ; NASM definition pseudo-instruction
```

Si encuentra que el código ensamblador confuso, hay una explicación más detallada aquí .

Breve explicación:

El número en EAX se refiere a una función del sistema. En este caso, el sistema es Linux y la llamada de sistema en EAX es 4. Linux sistema de llamada 4 se refiere a la función "sys_write", que escribe una cantidad dada de datos desde la memoria a un descriptor de archivo. Queremos escribir los datos en el "mensaje" dirección a la salida estándar, por lo que ponemos 1 en EBX porque 1 se refiere a la salida estándar. Luego ponemos la dirección de nuestro mensaje en ECX y la longitud del mensaje en EDX. Le estamos diciendo al sistema que tome ***longitud*** bytes de datos del ***mensaje de*** dirección y escribirlos en ***la salida estándar*** . En C, sys_write es una función con tres argumentos que tiene un prototipo de esta manera:

```
sys_write(unsigned int file_descriptor, const char * message, size_t length);
```

Una vez que hemos puesto nuestra syscall y sus argumentos en los registros adecuados, provocamos interrupción 80h. El código int 80h solicita al sistema que "hacerlo!"

La rutina sys_exit tiene una configuración similar. Ponemos la llamada al sistema sys_exit (1) en EAX y su único argumento (0) en EBX. Entonces llamamos a 80h de interrupción y el código en la dirección 80h de la tabla de vectores de interrupción se ejecuta sys_exit, limpiamente terminar nuestro programa.

Si el breve resumen no es suficiente para usted, considere revisar algunos de los enlaces en segundo plano.

3. Compilar el programa de la Asamblea en código máquina

Guarde su código ensamblador como hello32.nasm y emita el siguiente comando:

Sistema de 64 bits: `$ nasm-f elf64-o hello32.o hello32.nasm` 32 bits del sistema: `$ nasm-f elf-o hello32.o hello32.nasm` Esto se traducirá su programa conjunto en x86 código de máquina que su procesador pueda entender. NASM soporta una variedad de arquitecturas. Se puede ver la lista escribiendo `$ nasm-hf` Desafortunadamente, código de máquina en bruto no se puede ejecutar por sí mismo. Tiene que ser presentado en el sistema operativo en un formato normalizado con unas pocas piezas o metadatos relevantes. Linux utiliza el Formato ejecutable y enlazable (ELF) para ejecutables. Para ejecutar el código compilado, tenemos que vincularlo a un binario ELF.

4. Vincular el código de la máquina en un binario ELF

El siguiente comando utilizará ld , el enlazador dinámico y cargador, conectar nuestro código máquina compilado NASM en un binario ELF.

```
$ ld -o hello32 hello32.o
```

5. Ejecute el archivo ejecutable

El binario ELF recién vinculada se puede ejecutar como cualquier otro programa:

```
$ ./hello32
Hello World!
```

Si su código segfaulted antes o después de mostrar "Hello World", hay algo diferente entre su

sistema y el mío. Cualquiera de su sistema operativo y llamadas al sistema no son como la mía, su arquitectura no es como la mía, o si tiene un error en el código. Es posible que algo aún más extraño pasó, como una incompatibilidad en ld o NASM. Cualquier depuración se deja como ejercicio para el lector. Si el código no segfault, ¡enhorabuena!

Si eres como yo, puede haber escrito `$ wc hello32` y se dio cuenta de que su binario ELF es de alrededor de un kilobyte de tamaño. Probablemente esté pensando, "El código original es tan corto! Es todo lo que el espacio realmente necesario? "

Yo pensé lo mismo, y la respuesta es no. Con un poco de simplificar, podemos compilar nuestro código en un ejecutable ELF mucho menor .

Lectura adicional

Introducción a la programación en ensamblador UNIX

Manual NASM / Linux Asamblea Tutorial

Linux Tabla 2.2 Sistema de llamada / Linux Tabla 2.6.35 Sistema de llamada

Llamadas del sistema Linux para HLA Programadores por Randall Hyde (Este es el mejor trabajo sobre las llamadas del sistema Linux en el montaje que he encontrado)

Próximos Términos de búsqueda:

- NASM LINUX
- NASM TUTORIAL
- NASM HOLA MUNDO
- NASM LINUX
- HOLA NASM MUNDO
- NASM PARA LINUX
- NASM TUTORIAL LINUX
- NASM UBUNTU
- NASM EN LINUX
- NASM LD 64 32

Esta entrada fue archivada en **Programación** y etiquetada **Asamblea** , **hola mundo** , **linux** , **NASM** , **llamadas al sistema** , **tutorial** de **marca** . Guarda el **enlace permanente [http://blog.markloiseau.com/2012/04/hello-world-nasm-linux/]** .

5 thoughts on " "Hola Mundo" en Asamblea Linux 32-bit (NASM) "

fle4y
en**16 de noviembre 2012 a las 16:53** dijo:

agradable, thx!

Feliz
en**26 de marzo 2013 a las 11:30 am** dijo:

oh gracias yo estaba buscando una mejor explicación y esto me ayudó mucho.

Nikhil
en**28 de abril 2013 a las 2:00 pm** dijo:

Bro de Gr8 tut que funciona .....

Amelie
en**02 de mayo 2013 a las 20:26** dijo:

¿Qué hay de nuevo, reviso tu blog como cada semana. Su estilo de narración de historias es impresionante, mantener el buen trabajo!

Narasimhan
en**22 de julio 2013 a las 11:21 am** dijo:

Gracias una tonelada