**PROYECTO SYSCALL**

El objetivo de este proyecto es implementar un syscall en una versión de kernel específica, el cual debe tomar una cadena de caracteres con una expresión aritmética y debe indicar si está correctamente cerrada. Como ejemplo: {([2+2])} si está bien cerrada; {([2+2]) no está bien cerrada.

La distribución escogida para implementar el syscall fue Ubuntu 16.04 y Kernel 4.16.10

**¿Cómo funciona?**

Los system calls, o syscalls, son las únicas interfaces que el kernel de Linux usa para comunicarse con aplicaciones de usuario. Los programadores pueden usar recursos de kernel, incluyendo recursos de hardware llamando interrupciones directas o indirectas. Ahora, las aplicaciones generalmente usan los syscalls de kernel de manera indirecta, utilizando funciones de las librerías de C.

Generalmente los syscalls son hechos utilizando una forma funcional, permitiéndoles usar uno o más parámetros. El resultado de la ejecución de un syscall se representa con un valor de retorno. Se tiende a usar un valor negativo si se da un error, y el valor de 0 para indicar una ejecución correcta.

En el kernel de Linux, cada sistema tiene un número único de llamada de función de syscall. Esos números son definidos al inicio de la línea 62 en el archivo include/unistd.h. Por ejemplo, el syscall de escritura tiene una función de número 4.

Cuando una aplicación genera una interrupción INT 0x80 hacia el kernel vía una librería de función, un syscall es iniciado. El número del syscall es almacenado en el registro EAX, y los parámetros pueden ser almacenados en los registros EBX, ECX y EDX. Por lo tanto, el programador puede mandar hasta 3 parámetros directamente al kernel. Si el programador lo desea, también podría no mandar ningún parámetro. El proceso de manejar el syscall de interrupción INT 0x80 es el system\_call en el programa kernel/system\_call.s.

De manera a facilitar la ejecución de los syscalls, el código fuente del kernel define unos macros de funciones \_syscalln() en el archivo #include/unistd.h, con n representando el número de parámetros, cual puede ser entre 0 y 3. Si se necesita pasar grandes cantidades de información al kernel, se pueden usar los punteros de esa información.  
Por ejemplo, para el syscall de read(), su definición es:

int read(int fd, char \*buf, int n);

Si ejecutamos el syscall correspondiente en un programa de usuario, el macro del syscall será:

#define \_\_LIBRARY\_\_

#include<unistd.h>

\_syscall3(int, read, int, fd, char \*, buf, int, n)

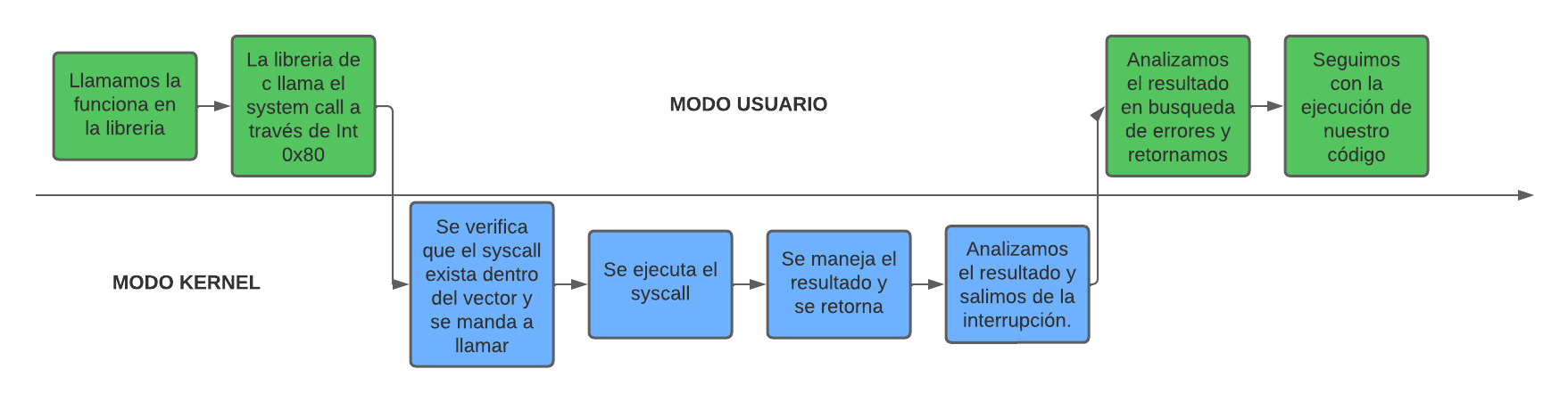
Se puede usar ese syscall3 directamente en un programa de usuario para ejecutar a un syscall de read() en vez de pasar por una librería de C. De hecho, la forma de la función llamada en una librería de C sería la misma que la dada aquí.

Para cada macro dado en el include/unistd.h, hay un total de 2+(2\*n) parametros. El primero corresponde al tipo de retorno que dará el system call, el segundo parámetro es el nombre del syscall, seguido del tipo y nombre del parámetro llevado por el syscall.

Si un system call requiere más de 3 parámetros, el kernel generalmente usa como parámetro al bloque regulador de parámetros y pasa el puntero del bloque como un parámetro al kernel. Por lo tanto, para los syscall con más de 3 parámetros, solamente se necesita usar el macro\_syscall1() con un argumento que manda el puntero del primer argumento al kernel. Por ejemplo, el select() tiene 5 argumentos, pero solamente se necesita mandar el puntero hacia su primer argumento.

Cuando se ingresa al la llamada de kernel kernel/sys\_call.s, el código del system\_call verifica primero si en ese syscall si el número de la función en el registro EAX es válido, es decir, si ese es incluido dentro de los números de system calls.

**Diagrama de System Call**

****

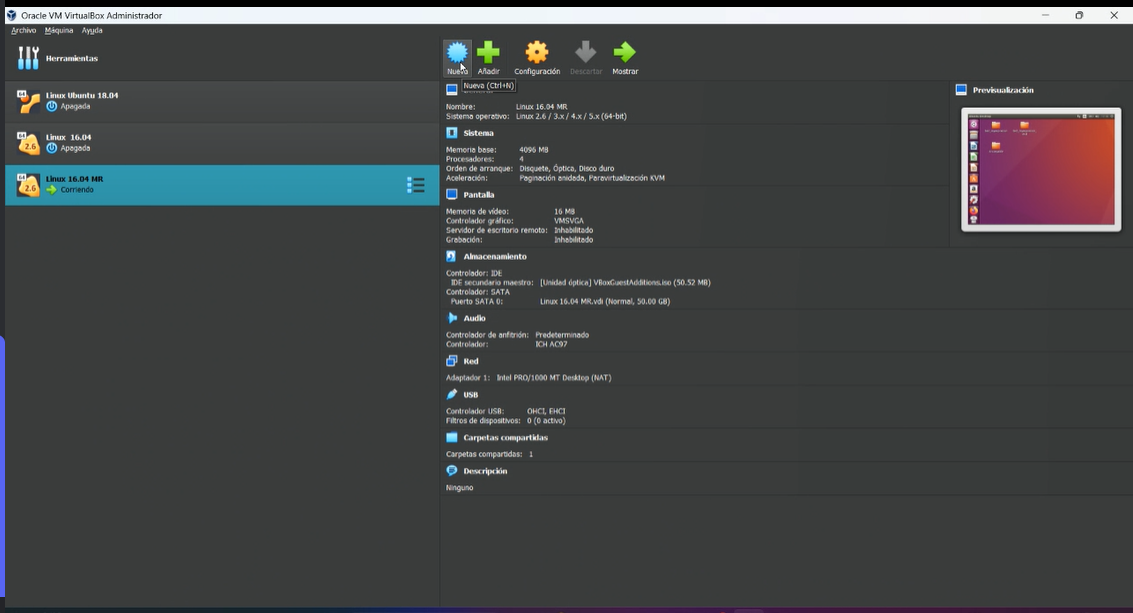
**INSTRUCCIONES DE IMPLEMENTACIÓN**

Prerequisitos:

* Instalar VirtualBox: [Downloads – Oracle VM VirtualBox](https://www.virtualbox.org/wiki/Downloads)
* Descargar la imagen de Ubuntu 16.04: [Downloads - Ubuntu 16.04](https://releases.ubuntu.com/16.04/)

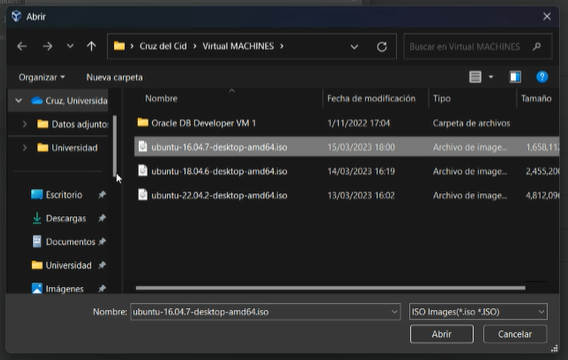
**Instrucciones para levantar la máquina virtual:**

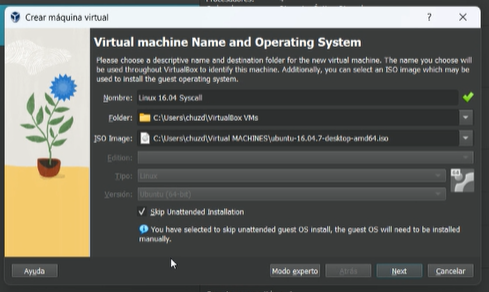
1. En el Virtualbox ir a NUEVA*.*

**

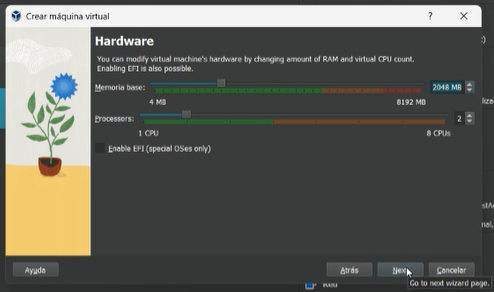
1. Ponerle un nombre a la máquina virtual, seleccionar dónde se guarda y seleccionar el ISO que se descargó (en este caso, Ubuntu 16). Después, presionar next.

Mantener SKIP seleccionado.

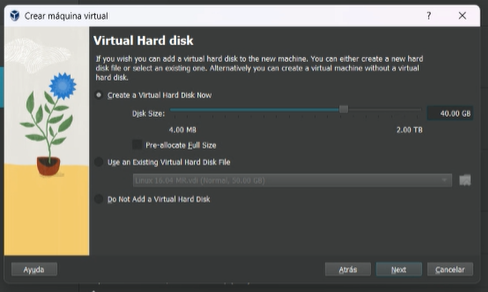




1. Asignar recursos:
   1. MÍNIMOS: 2GB de RAM y 2 núcleos
   2. RECOMENDADOS: 4GB de RAM y 4 núcleos
      1. Si el dispositivo tiene más capacidad, es mejor asignarle más memoría RAM y más núcleos, siempre dejándole a la máquina lo suficiente para seguir operando



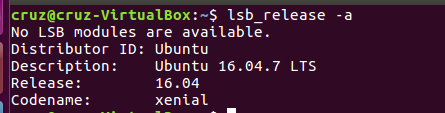
1. Asignarle 40GM de disco, ya que será necesario para compilar el Syscall y el SO



1. Una vez terminado, seleccionar la máquina recién creada y presionar INICIAR

**Instrucciones para crear el syscall:**

1. Abrir la terminal
2. Escribe el comando lsb\_release -apara saber que se está trabajando en la distribución correcta



1. Moverse a la carpeta /usr/src cd /usr/src

Ahí se encuentra el código fuente del kernel instalado actualmente.

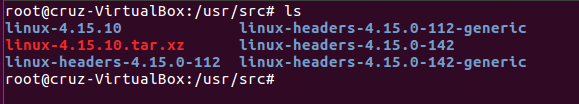
1. Entrar en modo administrador ejecutando sudo sue ingresando la contraseña del usuario que se creó al crear la computadora virtual.
2. Actualizar las dependencias del sistema operativo. Ejecutar apt-get update primero y después apt-get upgrade.
3. Instalar las siguientes librerías necesarias para la compilación del kernel:

apt-get install gcc libncurses5-dev bison flex libssl-dev libelf-dev gcc libncurses5-dev bison flex libssl-dev libelf-dev

1. Descargar la versión del kernel que se modificará; ejecutar el siguiente comando dentro de la carpeta que se especificó anteriormente:

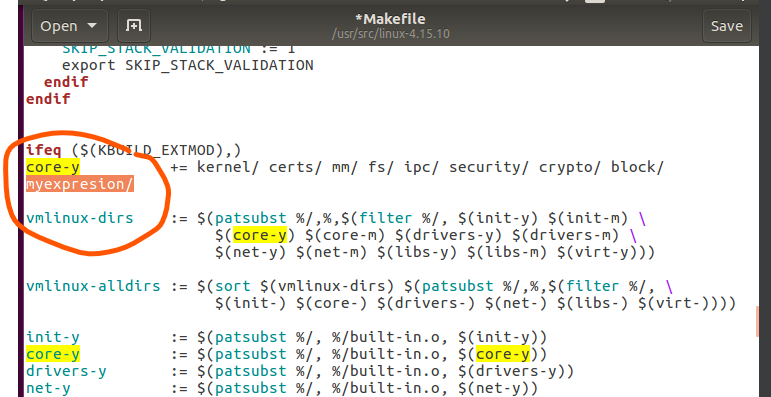
wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.15.10.tar.xz

1. Descomprimir el archivo: tar -xvf linux-4.15.10.tar.xz
2. Ejecutar lspara verificar que se descargaron todos los archivos



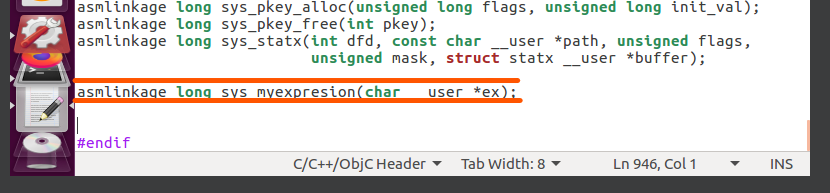
Se debe tener la carpeta *Linux-4.15.10*

1. Entrar a la carpeta *Linux-4.15.10*: cd linux-4.15.10
2. Crear la carpeta donde se añadirá el nuevo system call, como ejemplo está *myexpresion*: mkdir myexpresion
3. Crear un nuevo archivo llamado myexpresion.c: gedit myexpresion.c
4. Se abrirá una nueva ventana con una editor de texto. Copiar el código disponible en myexpresion.c dentro del repositorio y pegarlo en el editor. Presionar *save* y cerrar la ventana.
5. Crear el Makefile para compilar nuestro system call: gedit Makefile
6. En el editor de texto escribir:
   * *obj-y := myexpresion.o*
   * Presionar *save* y cerrar la ventana.
7. Salir a la carpeta linux-4.15.10 : cd ..
8. Ahora modificar el Makefile del kernel para que entre y compile el nuevo syscall
   * gedit Makefile
   * Buscar *core-y* y añadir la carpeta al final y color */*, como se muestra a continuación:



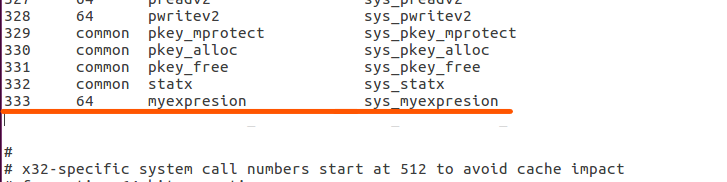
* + Presionar *save* y cerrar la ventana.

1. Crear el prototipo del nuevo syscall:
   * Ejecutar: gedit include/linux/syscalls.h, esto abrirá el documento con las definiciones de los syscalls en el editor de texto.
   * Avanzar hasta el final del documento y agregar la siguiente definición antes del *#endif:*

**

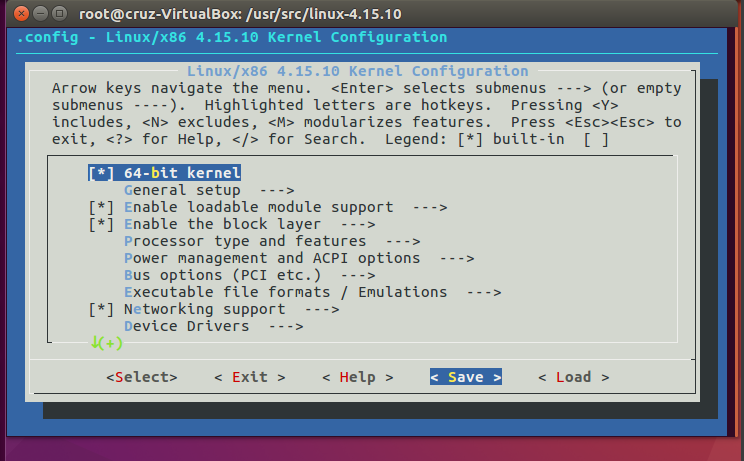
* + Presionar *save* y cerrar la ventana.

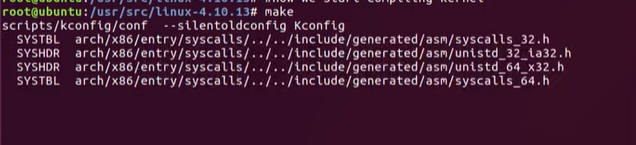
1. Añadir el nuevo syscall a la tabla de syscalls para poder ser llamado:
   * Ejecutar gedit arch/x86/entry/syscalls/syscall\_64.tbl
   * Avanzar hasta el índice 332 en la tabla de syscalls y agregar el nuevo syscall



* + Colocar en el índice 333 al nuevo system call
  + Presionar *save* y cerrar la ventana.

1. Regresar a la carpeta Linux-4.15.10 y ejecutar el siguiente comando: make menuconfig
2. Dentro de ese editable, presionar *save* para guardar la configuración y después salir



1. Ejecutar el comando make oldconfig para guardar la configuración anterior.
2. Por último se compila el Kernel
   * Ejecutar make --jobs 2
   * El comando anterior se puede modificar. Si se le asignaron más CPU’s a la máquina virtual, se puede colocar el número de CPU’s que se asignaron para que la compilación sea más rápida. Si solo se quiere usar uno se puede colocar solo el comando make sin banderas.
3. Al finalizar la configuración ejecutar shutdown -r now para reiniciar la máquina y aplicar todos los cambios.

**Prueba del System Call:**

Para probar el system call se debe descargar la carpeta *test\_myexpresion*. Ya dentro de la carpeta abrir la terminal y ejecutar gcc test\_myexpresion.c para compilar. Al finalizar la compilación, ejecutar el programa con ./a.out**.** Al ingresar la expresión para evaluar retornará verdadero o falso. También se puede ver el output del system call ejecutando sudo dmesg.