Laboratorio: Compilar y correr una aplicación sin SO

Parte 1: Clonar el repositorio completo

Parte 2: Compilar y ejecutar los ejemplos

Parte 3: Grabar la imagen y correrlas en hw real

Aspectos básicos/Situación

En este laboratorio estaremos solos frente al hardware, no existirá un sistema operativo que nos de soporte. Tendremos control total y absoluto del procesador. Los procesadores x86 tal y como los conocemos pueden correr el mismo código que corrían sus antepasados mas de 30 años atrás. Sin embargo los procesadores fueron evolucionando y agregando nuevas capacidades año a año. Para acceder a las mismas es necesario configurar ciertos aspectos del microprocesador. Al finalizar este laboratorio habrá realzado el primer salto de evolución del procesador pasándolo de modo real a modo protegido.

Recursos necesarios

1. Computadora con so linux preferentemente ububtu.
2. VS Code u otro editor de su agrado

Instrucciones

Parte 1 – Clonar el repositorio Git y el submódulo

En esta parte, clonará un repositorio de gitlab para comenzar a trabajar. A continuación podrá clonar el submódulo.

Paso 1: Clonar el repositorio.

Con la siguiente instrucción podrá tener el repositorio de este laboratorio.

```
javier@javier-ThinkPad-T450: ~$ git clone
https://gitlab.unc.edu.ar/javierjorge/protected-mode-sdc.git
```

Paso 2: Trabajando con submódulos.

Los submódulos permiten vincular un repositorio dentro de otro. Esto se hace con un comando especial, **git submodule add,** este comando creará una entrada en un archivo dentro del repositorio. Esta tarea ya fue realizada por el instructor. Puede ver el contenido del archivo ejecutando:

```
javier@javier-ThinkPad-T450: $ cd protected-mode-sdc
javier@javier-ThinkPad-T450: $ cat .gitmodules
javier@javier-ThinkPad-T450: $ ls x86-bare-metal-examples/
```

Habrá podido observar que la carpeta esta vacía. Para inicializar y clonar el repositorio ya incluido como submódulo debe ejecutar los siguientes comandos:

```
javier@javier-ThinkPad-T450: $ git submodule init
javier@javier-ThinkPad-T450: $ git submodule update
javier@javier-ThinkPad-T450: $ ls x86-bare-metal-examples/
javier@javier-ThinkPad-T450: $ cd x86-bare-metal-examples/
```

Laboratorio: Compilar y correr una aplicación sin SO

Parte 2: Compilar y ejecutar los ejemplos

Paso 1: instalar qemu, compilar y ejecutar los ejemplos

Primero debe instalar qemu y luego, una vez extraídos los ejemplos puede empezar a ejecutarlos utilizando un script incluido llamado “run”. También es recomendable que tenga instalado gdb-dashboard

```
javier@javier-ThinkPad-T450: $ sudo apt install qemu-system-x86
javier@javier-ThinkPad-T450: $ ./run bios_hello_world
```

Este script se encarga de complilar el ejemplo bios_hello_world.S y correrlo. Puede modificar el texto del mensaje con su editor favorito y volver a correrlo.

```
javier@javier-ThinkPad-T450: $ ./run protected_mode
```

Hará lo mismo con el ejemplo protected_mode.S. Puede modificar el texto del mensaje con su editor favorito y volver a correrlo.

Paso 2: Depurar los ejemplos

Puede depurar los ejemplos usando gdb, es recomendable que instale la extensión gdb dashboard para mejorar la visualización.

```
javier@javier-ThinkPad-T450: $ ./run bios_hello_world debug
```

Parte 3: Grabar la imagen y correrla en HW real

Para grabar la imagen no hay nada mejor que el comando dd. Con el podrá realzar una copia bit a bit de una imagen de disco a un disco real. En este caso necesitaremos un pendrive para grabarle los primeros 512bytes. Recuerde resguardar su información antes de proceder.

Paso 1: Determinar cual es el driver asignado al dispositivo

para ello usará el comando fdisk en combinación con grep para listar solamente las unidades de tipo disco.

```
javier@javier-ThinkPad-T450: $ sudo fdisk -l | grep sd
[sudo] password for javier:

Disk /dev/sda: 465,8 GiB, 500107862016 bytes, 976773168 sectors
/dev/sda1  *         2048   1126399   1124352   549M  7 HPFS/NTFS/exFAT
...
/dev/sda6       514455552 968959999 454504448 216,7G 83 Linux
Disk /dev/sdb: 7,5 GiB, 8004304896 bytes, 15633408 sectors
/dev/sdb1       2422640742 4845281483 2422640742  1,1T 66 unknown
..
/dev/sdb4       2422640742 4845281483 2422640742  1,1T 66 unknown
```

Busque el dispositivo del tamaño apropiado. Quitelo de manera segura verifique que el dispositivo ya no está. Vuelva a conectar el dispositivo y verifique que mantiene la misma denominación. En este caso es un pendrive de 8 Gb y esta reconocido en este sistema en el archivo `/dev/sdb usted deberá determinar su propio archivo.`

Paso 2: Grabar la imagen de disco

Laboratorio: Compilar y correr una aplicación sin SO

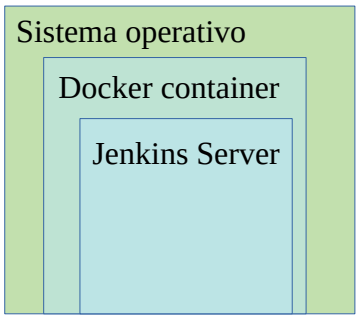
Podrá grabar la imagen usando el comando dd. ATENCION: NO USE DD SOBRE SU DISCO PRINCIPAL

```
javier@javier-ThinkPad-T450: $ sudo dd if=protected_mode.img of=/dev/sdX
```

Una vez grabada la imagen en el pendrive es necesario reiniciar la PC e indicar a la bios que debe arrancar desde el dispositivo usb. Con ello bastará para poder ver este software primitivo correr en hw real. Si va a retirar el pendrive hagalo de manera segura para que todas las operaciones se realicen de forma efectiva.

Laboratorio: Compilar y correr una aplicación sin SO

Paso 4: Investigar los niveles de abstracción que se están ejecutando actualmente en el equipo.



Parte 3: Configurar Jenkins

En esta parte, se completará la configuración inicial del servidor Jenkins.

Paso 1: Abrir una pestaña del navegador.

Vaya a http://localhost:8080/ e inicie sesión con su contraseña de administrador copiada.

Paso 2: Instalar los complementos de Jenkins recomendados.

Haga clic en **Install suggested plugins** y espere a que Jenkins descargue e instale los complementos. En la ventana del terminal, verá mensajes de registro a medida que la instalación continúe. Asegúrese de no cerrar esta ventana de terminal. Puede abrir otra ventana de terminal para acceder a la línea de comandos.

Paso 3: Omitir la creación de un nuevo usuario administrador.

Una vez finalizada la instalación, se le mostrará la ventana **Create First Admin User**. Por ahora, haga click en **Skip and continue as admin** en la parte inferior.

Paso 4: Omitir la creación de una instancia.

En la ventana **Instance Configuration**, no cambie nada. Haga clic en **Save and Finish** en la parte inferior.

Paso 5: Empezar a usar Jenkins.

En la siguiente ventana, haga clic en **Start using Jenkins**. Ahora debería estar en el panel principal con un mensaje de **Welcome to Jenkins!** .

Parte 4: Usar Jenkins para ejecutar una compilación de la aplicación

La unidad fundamental de Jenkins es el **job** (también conocido como un **project**). Puede crear **jobs** que realicen una variedad de tareas, entre las que se incluyen las siguientes:

- Recuperar código de un repositorio de administración de código fuente como GitHub.
- Compilar una aplicación utilizando una secuencia de comandos (**script**) o una herramienta de compilación.
- Empaquetar una aplicación y ejecutarla en un servidor

En esta parte, creará un **job** de Jenkins simple que recupera la última versión de su aplicación de muestra de GitHub y ejecuta el script de compilación. En Jenkins, puede probar su aplicación (**Parte 5**) y agregarla a una **pipeline** de desarrollo (Parte 8).

Laboratorio: Compilar y correr una aplicación sin SO

Paso 1: Crear un nuevo job.

a. Haga click en el enlace **Create a job** directamente debajo del mensaje: **¡Welcome to Jenkins!** También puede hacer clic en **New Item** en el menú de la izquierda.
b. En el campo **Enter an Item Name** ponga como nombre **BuildAppJob**.
c. Haga click en **Freestyle Project** como tipo de trabajo. En la descripción, la abreviatura de ACS significa administración de configuración de software (Software Configuration Management, SCM), que es una clasificación de software responsable del seguimiento y control de los cambios en el software.
d. Arrástrelo hasta la parte inferior de la página y haga click en **OK**.

Paso 2: Configurar el Jenkins BuildAppJob.

Ahora se encuentra en la ventana de configuración donde se pueden introducir detalles sobre su trabajo. Las pestañas en la parte superior son accesos directos a las secciones siguientes. Hacer clic en las pestañas para explorar las opciones que se pueden configurar. Para este trabajo, sólo se necesita agregar algunos detalles de configuración.

a. Hacer clic en la pestaña **General** y agregar una descripción para su trabajo. Por ejemplo, "**Mi primer trabajo en Jenkins.**"
b. Hacer clic en la pestaña **Administración de código fuente (Source Code Management)** y elegir el botón de opción **Git**. En el campo URL del repositorio, agregue el enlace del repositorio de GitHub para la aplicación de muestra, teniendo en cuenta que su nombre de usuario distingue mayúsculas de minúsculas. Asegúrese de agregar la extensión .git al final de su URL.

   Por ejemplo:

   `https://github.com/`*`github-username`*`/calculadora.git`

c. Para **Credenciales (Credentials)**, haga clic en el botón **Agregar (Add)** y elija **Jenkins**.
d. En el cuadro de diálogo **Agregar credenciales (Add Credentials)**, rellene con su nombre de usuario y contraseña de GitHub y, a continuación, haga clic en **Agregar (Add)**.

   Nota: Recibirá un mensaje de error que indica que la conexión ha fallado. Esto se debe a que aún no ha seleccionado las credenciales.

e. En el menú desplegable de **Credenciales (Credentials)** donde actualmente dice **Ninguno (None)**, elija las credenciales que acaba de configurar.
f. Después de agregar la URL y las credenciales correctas, Jenkins prueba el acceso al repositorio. No debería haber mensajes de error. Si los hay, verificar la URL y las credenciales. Tendrá que **añadirlos** de nuevo, ya que en este momento no hay forma de eliminar los que se han introducido anteriormente.
g. En la parte superior de la ventana de configuración de **BuildAppJob**, haga clic en la pestaña **Build**.
h. Abra menú desplegable **Add build step** y elija **Execute shell**.
i. En el campo **Command**, escriba el comando que utiliza para ejecutar la compilación make.

   ```
   make
   ```

j. Haga clic en el botón **Save**. Será regresado al panel de Jenkins con **BuildAppJob** seleccionado.

Nota: Como el contenedor no tiene acceso a las aplicaciones del host, es necesario actualizar la lista de paquetes disponibles y luego instalar make y gcc. La forma mas sensilla en sistemas debian es instalar el pakete build-essential.

k. Primero debe acceder al docker mediante una consola para poder instalar los paketes.

Laboratorio: Compilar y correr una aplicación sin SO

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker exec -it
jenkins_server /bin/bash
```

l. Luego instalar los paquetes necesarios

```
root@f2c9059c2cea:/# apt update && apt install build-essential
root@f2c9059c2cea:/# apt clean
root@f2c9059c2cea:/# rm -rf /var/lib/apt/lists/*
```

m. Puede salir con **exit**

n. Para que la proxima vez que intente levantar la instancia estos cambios sean permantentes se puede crear una nueva instancia de este docker con los cambios realizados usando el comando commit.

```
javier@javier-ThinkPad-T450:~$ docker ps -l
CONTAINER ID        IMAGE                 COMMAND
CREATED             STATUS              PORTS
NAMES
918dcdd85e8c        jenkins/jenkins:lts   "/sbin/tini -- /usr/…"   9
← javier@javier-ThinkPad-T450:~$ docker commit 918dcdd85e8c jenkins-build-
essentials
```

p. Detener la imagen original

```
← javier@javier-ThinkPad-T450:~$ docker container stop jenkins_server
```

r. La proxima vez debera correr la nueva imagen...

```
docker run --rm -u root -p 8080:8080 -v
jenkins-data:/var/jenkins_home -v $(which docker):/usr/bin/docker -v
/varun/docker.sock:/var/run/docker.sock -v "$HOME":/home --name
jenkins_server jenkins-build-essentials
```

Paso 3: Utilizar Jenkins para compilar una aplicación.

En la parte izquierda, haga click en **Build Now** para iniciar el trabajo. Jenkins descargará su repositorio de Git y ejecutará el comando de compilación make. Su compilación debería tener éxito porque no ha cambiado nada en el código desde la Parte 3 cuando modificó el código.

Paso 4: Acceder a los detalles de la compilación.

A la izquierda, en la sección **Build History**, haga click en su número de compilación, que debería ser el **#1**, a menos que haya compilado la aplicación varias veces.

Paso 5: Ver la salida de consola.

A la izquierda, haga click en **Console Output**. Debe ver un resultado similar a lo siguiente. Observe los mensajes de éxito en la parte inferior, así como la salida del comando **docker ps -a**. Se están ejecutando al menos un contenedor docker para Jenkins en el puerto local 8080.

```
Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/BuildAppJob
The recommended git tool is: NONE
using credential 66b4a183-1951-49fc-a9a7-e06646aa9f26
Cloning the remote Git repository
Cloning repository https://github.com/MkSolinas/calculadora.git
 > git init /var/jenkins_home/workspace/BuildAppJob # timeout=10
Fetching upstream changes from https://github.com/MkSolinas/calculadora.git
 > git --version # timeout=10
```

Laboratorio: Compilar y correr una aplicación sin SO

```
 > git --version # 'git version 2.20.1'
using GIT_ASKPASS to set credentials
 > git fetch --tags --force --progress -- https://github.com/MkSolinas/calculadora.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/MkSolinas/calculadora.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision b22a57adb836500f475a3a5ccb55961b671e64c4 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f b22a57adb836500f475a3a5ccb55961b671e64c4 # timeout=10
Commit message: "Confirmando archivos de aplicación ejemplo"
First time build. Skipping changelog.
[BuildAppJob] $ /bin/sh -xe /tmp/jenkins8608235895889754566.sh
+ make
make: 'calc' is up to date.
Finished: SUCCESS
```

Laboratorio: Compilar y correr una aplicación sin SO

Parte 5: Usar Jenkins para testear una compilación

En esta parte agregaremos un segundo **job** para testear la aplicación y asi poder asegurarse de que funciona correctamente.

Paso 1: Editar el trabajo para probar su aplicación de muestra.

a. Vuelva a la pestaña del navegador web Jenkins y haga clic en el enlace **Jenkins** en la esquina superior izquierda para volver al panel principal.

b. Haga click en el enlace **del nombre del job** debajo de la etiqueta “name” para acceder al trabajo anterior.

c. Haga click en el enlace “configure” en el panel izquierdo.

Paso 2: Configurar Jenkins para correr una prueba.

a. Haga clic en la pestaña **Build**.

b. Haga clic en **Add build step** y elija **Execute shell**.

c. Introducir el nombre del script de pruebas ya incluido ./test.sh Este escript intenta realizar una serie de operaciones. Si las operaciones devuelven el resultado esperado, el script sale con un código 0 (cero), lo que significa que no hay errores en la compilación en el Job. Si falla algun assert, el script sale con un código de 1 (uno), lo que significa que el Job falló.

d. Guarde los cambios

Paso 4: Hacer que Jenkins ejecute el trabajo BuildAppJob de nuevo.

a. Actualice la página web con el botón de actualizar de su navegador.

b. Haga clic en el botón de compilación en el extremo derecho (un reloj con una flecha).

Paso 5: Verificar que hayan finalizado correctamente

Si todo va bien, deberíamos ver la marca de tiempo para la actualización de la columna **Último éxito** para **BuildAppJob** y **TestAppJob**. Esto significa que el código para ambos trabajos se ejecutó sin errores. Pero también podemos verificar esto por nosotros mismos.

Nota: Si las marcas de hora no se actualizan, asegúrese de activar la actualización automática haciendo clic en el vínculo situado en la esquina superior derecha.

a) Haga clic en el vínculo de la última compilación y, por consiguiente, haga clic en **Salida de consola (Console Uotput)**. Debe ver un resultado similar al siguiente:

```
Iniciado por el proyecto
...
+ exit 0 Terminado:
EXITOSO
```

Paso 6: Detener el contenedor docker

Con el siguiente comando para ver el nombre del container que está corriendo y detenerlo:

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker container ls
```

Laboratorio: Compilar y correr una aplicación sin SO

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker container stop
jenkins_server
```