Instalación de Pintos y guía de pruebas

David Flores Peñaloza (dflorespenaloza@gmail.com)
Jorge Luis García Flores (jorgel_garciaf@ciencias.unam.mx)
Angel Renato Zamudio Malagón (renatiux@gmail.com)

1. Instalación

Pintos es un sistema operativo de propósito educativo desarrollado por la universidad de Stanford. Está diseñado para operar sobre una arquitectura x86 de 32 bits, está desarrollado mayormente en lenguaje C pero tiene componentes en ensamblador. Aunque Pintos es capaz de ejecutarse sobre una arquitectura física que cumpla las características antes mencionadas, se prefiere utilizar una máquina virtual con el propósito de agilizar el proceso de desarrollo y pruebas. Pintos está diseñado para trabajar con cualquier máquina virtual sin embargo para agilizar el proceso de compilación, empaquetado y ejecución, cuenta con una serie de scripts que sirven para dos máquinas virtuales en particular: Qemu y Bochs.

1.1. Instalación de Qemu (desde repositorios)

Qemu es una máquina virtual de código abierto que emula varias arquitecturas, en particular la arquitectura x86. La forma más sencilla de instalar Qemu es por medio de los repositorios de la distribución de Linux que utilizamos. A continuación se muestra la forma de hacerlo para las distribuciones más populares:

- **Debian/Ubuntu**: apt-get install qemu
- **Arch**: Pacman -S qemu
- **Fedora**: dnf install @virtualization
- **Gentoo**: emerge --sk app-emulation/qemu
- **Centos**: yum install qemu-kvm
- **SUSE**: zypper install qemu

En muchos casos la instalación instala ejecutables para cada una de las arquitecturas, sin embargo Pintos requiere que se pueda acceder al emulador de la arquitectura x86 desde el comando **qemu**, por ello necesitamos crear una liga simbólica de la siguiente forma:

ln -s /usr/bin/qemu-system-i386 /usr/bin/qemu

1.2. Instalación de Qemu (código fuente)

Si por alguna razón no es posible instalar Qemu por medio de los repositorios, tenemos la opción de instalar por medio del código fuente. Este proceso es un poco más complicado pues debemos compilar el código fuente de Qemu.

Para la compilación es necesario tener una versión del compilador GCC instalado, las versión del compilador depende de la versión de Qemu, para ello se recomienda revisar la documentación en el sitio oficial https://qemu.org. Adicionalmente, las siguientes bibliotecas son requeridas:

- libpixman-1-dev
- libsdl1.2-dev

El siguiente paso es descargar el código fuente de la maquina virtual, para ello lo hacemos desde la página oficial y bajamos la versión estable más reciente:

http://qemu.org

Una vez que hemos bajado el código fuente, lo extraemos y colocamos en la ruta donde se quedará de forma permanente. Después ingresamos al directorio principal del código fuente y ejecutamos el siguiente comando:

./configure --target-list=i386-softmmu

Con lo anterior estamos configurando Qemu para que únicamente se compile la versión i386 (x86) de la máquina virtual, pues es la única que necesitamos para Pintos. Por último compilamos el código fuente ejecutando el siguiente comando:

make

Para poder acelerar el proceso de compilación de puede incluir el parámetro *-jN* donde se sustituye *N* por el número de threads que se desea utilizar para compilar; por ejemplo si queremos utilizar 4 threads para la compilación el comando sería: **make -j4**. Después de compilar el código fuente es necesario que el ejecutable de la máquina virtual se encuentre disponible desde cualquier directorio; para ello ejecutamos, con permisos de administrador, el siguiente comando:

ln -s /ruta_de_qemu/i386-softmmu/qemu-system-i386 /usr/bin/qemu

1.3. Instalación de Pintos

El proceso de instalación de Pintos se reduce a compilar el código fuente y verificar que Qemu puede ejecutar sin problemas el sistema operativo. Primero debemos obtener el código fuente, para ello entramos a la siguiente página y bajamos el archivo **pintos.tar.gz**:

https://sites.google.com/view/sistemasoperativos20192/laboratorio

Después descomprimimos el archivo *tar* en cualquier directorio al que se tenga acceso (el *home* del usuario es buena opción). Ahora necesitamos que los *scripts* del directorio *src/utils* sean accesibles desde cualquier ubicación en la terminal; hay varias formas de hacerlo pero la más sencilla y menos agresiva es agregar la ruta a la variable *PATH*; para hacerlo solamente para nuestro usuario tenemos que agregar la siguiente línea al archivo .bashrc, que se encuentra en el directorio *home* de nuestro usuario:

export PATH=$PATH:/home/nuestro_usuario/ruta_pintos/src/utils

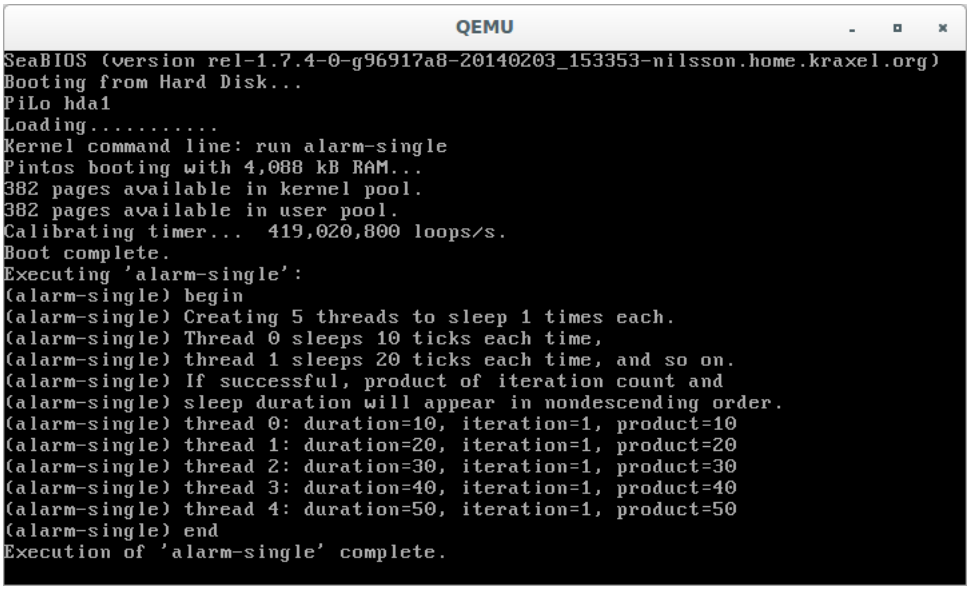
Es necesario cerrar y volver a abrir las terminales que estemos usando para que los cambios se vean reflejados. Ahora procedemos a compilar, para ello debemos ingresar en el directorio *src/threads* y desde ahí ejecutar el comando:

make

Si el proceso de compilación fue exitoso, debe existir un nuevo directorio dentro de *src/threads* llamado *build*, en este directorio se encuentra el kernel de Pintos compilado así como también un directorio (*tests*) donde se guarda registro de las pruebas que se han ejecutado. Para verificar que la compilación se realizó de forma satisfactoria ejecutamos el siguiente comando:

pintos run alarm-single

Si todo se realizó de forma correcta, debe de aparecer la ventana del emulador Qemu, el cual comenzará a ejecutar Pintos, y después de proceso de *booting* comenzará a ejecutar la prueba *alarm-single*. A continuación se muestra un ejemplo de una ventana de Qemu ejecutando Pintos y la prueba *alarm-single*:



También es necesario probar que Pintos esté apagando correctamente la máquina virtual, pues las pruebas registran la salida de la ejecución después de que la máquina virtual se haya apagado. Para ello es suficiente ejecutar la misma prueba que anteriormente ejecutamos pero con un parámetro extra, de la siguiente forma:

pintos -- -q run alarm-single

Si todo es correcto, la ventana debe aparecer justo como en la ejecución anterior, pero ahora, después de que se termine la ejecución de la prueba la ventana debe cerrarse. Si esto no ocurre, manda un correo al ayudante.

2. Pruebas

Pintos cuenta con una serie de pruebas que están diseñadas para evaluar los requerimientos que el alumno tiene que implementar. Las pruebas son programas escritos en lenguaje C y se dividen en dos tipos, las que se ejecutan en modo kernel y las que se ejecutan como procesos de usuario. Las pruebas que se ejecutan en modo kernel son aquellas que corresponden al primer proyecto (*threads*). El código fuente de las pruebas se encuentra en el directorio *src/tests* y se organizan en sub-directorios nombrados en función del nombre del proyecto al que pertenecen.

Es posible ejecutar las pruebas de dos formas, la primera solamente se ejecuta la prueba y se muestra la salida, en la segunda forma aparte de ejecutar la prueba es posible verificar si dicha prueba pasó o falló. Para ejecutar una prueba sin verificar si pasó o no, es necesario ejecutar el siguiente comando:

pintos run nombre_de_la_prueba

En el comando anterior se debe sustituir *nombre_de_la_prueba* por el nombre de la prueba que se desea ejecutar. Es necesario señalar que el comando anterior solamente es válido para las pruebas del primer proyecto, pues para los demás proyectos en necesario crear un disco virtual y copiar los ejecutables dentro del mismo para poder ejecutar las pruebas como procesos de usuario. En la sección correspondiente al primer requerimiento del segundo proyecto se mostrará como realizar dicha ejecución. La segunda forma de ejecutar las pruebas se realiza mediante la ejecución del siguiente comando:

make build/tests/nombre_proyecto/nombre_prueba.result

Cuando la prueba termina de ejecutarse se muestra en la consola si la prueba pasó (*pass*) o si falló (*FAIL*), adicionalmente se generan tres archivos:

1. **prueba.output:** Contiene la salida estándar que generó la prueba. Si se desea volver a ejecutar la prueba, es necesario borrar éste archivo.
2. **prueba.errors:** Contiene líneas adicionales que notifica la máquina virtual. Pueden contener posibles errores.
3. **prueba.result:** En caso de que la prueba se haya ejecutado satisfactoriamente, el archivo contiene la palabra *PASS*), en caso contrario, contiene una descripción de la salida esperada junto con las diferencias con la salida que produjo la prueba.

Si se quiere ejecutar todas las pruebas del proyecto en cuestión se puede hacer mediante el siguiente comando. Sin embargo hay que señalar que ejecutar todas las pruebas puede ser un proceso tardado.

make check

El comando anterior aparte de ejecutar todas las pruebas y mostrar una por una si pasó o falló, también genera un listado en el cual se pueden verificar el resultado de cada una de las pruebas. El listado se encuentra en el archivo *src/proyecto/build/tests/result*.