Carlos Alejandro Montiel Lorenzana  
15000552

00 – Iniciar PintOS

**1. Cual es la diferencia entre los simuladores Bochs y QEMU**

La principal diferencia de estos dos emuladores de hardware virtuales es la amplitud de dispositivos x86 que pueden ser emulados, en qemu podemos emular lo siguiente:

* CD-ROM/DVD-drive using an ISO image
* Floppy disk
* ATA controller or Serial ATA AHCI controller
* Graphics card (Cirrus CLGD 5446 PCI VGA-card,Standard-VGA graphics card with Bochs-VESA-BIOS-Extensions - Hardware level, including all non-standard modes, and an experimental patch that can accelerate simple 3D graphics via OpenGL, Red Hat QXL VGA or VirtIO GPU)
* Network card (Realtek 8139C+ PCI network adapter)
* Depending on target: i82551, i82557b, i82559er, NE2000 PCI, NE2000 ISA, PCnet, RTL8139, e1000 (PCI Intel Gigabit Ethernet), SMC91c111, Lance and mcf\_fec.[27][28]
* NVMe disk interface
* Parallel port
* PC speaker
* i440FX/PIIX3 (PCI and ISA) or Q35/ICH9 (PCIe and LPC) chipsets
* PS/2-mouse and -keyboard
* SCSI controller (LSI MegaRAID SAS 1078, LSI53C895A, NCR53C9x as found in the AMD PCscsi and Tekram DC-390 controllers)
* Serial interface
* Sound card (Sound Blaster 16, AudioPCI ES1370, Gravis Ultrasound, AC97, and Intel HD Audio[29])
* Watchdog timer (Intel 6300 ESB PCI, or iB700 ISA)
* USB 1.x/2.x/3.x controllers (UHCI, EHCI, xHCI)
* USB devices: audio, Bluetooth dongle, HID (keyboard/mouse/tablet), MTP, serial interface, CAC smartcard reader, storage (bulk-only transfer and USB Attached SCSI), Wacom tablet
* Paravirtualized VirtIO devices: block device, network card, SCSI controller, serial interface, balloon driver, 9pfs filesystem driver
* Paravirtualized Xen devices: block device, network card, console, framebuffer and input device
* The BIOS implementation used by QEMU starting from version 0.12 is SeaBIOS. The VGA BIOS implementation comes from Plex86/Bochs.The UEFI firmware for QEMU is OVMF.

Por otra parte Bochs habla la misma arquitectura que qemu, bochs puede emular lo siguiente,

* Video card Cirrus Logic CL-GD5430 ISA
* Cirrus Logic CL-GD5446 PCI
* Sound card Sound Blaster 16 card (ISA, no Plug & Play), Basic Sound Device
* Ethernet network card NE2000 (ISA/PCI) Ethernet or Intel(R) 82540EM Gigabit Ethernet adapter (PCI)[6]
* Chipset Intel 440FX PCI. Host-to-PCI bridge (PMC/DBX), PCI-to-ISA bridge, PCI IDE controller (PIIX3) are available. For PCI cards there are 5 PCI slots.
* USB Root hub and the devices mouse (optional), tablet, keypad (default), disk.
* SMP Can simulate up to 8 CPUs.
* Enhanced BIOS or SeaBIOS ElTorito, EDD v3.0, basic APM, PCIBIOS features, PCI interrupt routing table. 32-bit init for ACPI, SMM and SMP.

**2. Que simulador eligio para correr pintos?**

Los dos llevan a un software muy complejo y la incertidumbre de cual se puede usar en el proyecto se responde en que tanto se va a hacer, hasta este punto yo elegi qemu ya que encontre un poco mas de documentacion que bochs y fue mas accesible su informacion

<https://tssurya.wordpress.com/2014/08/16/installing-pintos-on-your-machine/>

**3. Que dificultad encontro al correr pintOS?**

La dificultad se basa en generar configurar todo el kernel y sus funciones para que pueda accesar la linea de comandos al comando pintos, todavia sigo teniendo un problema que es un loop infinito pero no se en donde esta, estas son las instrucciones que yo use para instalar pintos con qemu.

Otra dificultad es que los links de C a sus ibrarias nativas eran imposibles accederlas, no entiendo porque en mi sistema operativo Linux 19.1 Tessa tuvo ese gran problema, se resolvio instalando libappindicator-dev con la siguiente instruccion

run sudo apt-get install libappindicator-dev

# INSTRUCCIONES DE INSTALACION DE PINTOS

open terminal write echo ~

to see if your $HOME variable is ='/home/user-name' where user=name is your username

run sudo apt-get install qemu

run sudo apt-get install libappindicator-dev

go to $HOME and create a new folder named "os-pintos"

go to /home/user-name/os-pintos/pintos/src/utils/pintos-gdb

find the variable GDBMACROS and change that code line as below

GDBMACROS=/home/user-name/os-pintos/pintos/src/misc/gdb-macros

Now go to $HOME/os-pintos/pintos/src/utils/Makefile

change the line with LDFLAGS =-lm to LDLIBS=-lm

go to your terminal and browse the ../utils directory and type make it

will compile everything you need.

Now you have to edit make.vars

go to $HOME/os-pintos/pintos/src/threads/Make.vars

change the last line or the line number 7 with from SIMULATOR=--bochs to SUMULATOR=-qemu

assuming you are in the threads directory type make to compile everything

Make these three changes to the /pintos file in “$HOME/os-pintos/pintos/src/utils/” by opening it using some editor.

a. Change line no. 103 to “$sim = “qemu” if !defined $sim;” to enforce qemu as simulator.

b. On line no. 259, replace “kernel.bin” to path pointing to kernel.bin file at “$HOME/os-pintos/pintos/src/threads/build/kernel.bin”.

i.e the line should be ” my $name = find\_file (“/home/linux/os-pintos/pintos/src/threads/build/kernel.bin”);” where $HOME=/home/linux/ in my case.

c. Change line no. 623 to” my (@cmd) = (‘qemu-system-x86\_64’);”

Now we gonna edit the pintos.pm file in the same directory

Open “Pintos.pm” file present in “$HOME/os-pintos/pintos/src/utils/Pintos.pm” using a text editor and make the following change.

Replace “loader.bin” at line no.362 to the path “$HOME/os-pintos/pintos/src/threads/build/loader.bin”.

i.e is “$name = find\_file (“/home/linux/os-pintos/pintos/src/threads/build/loader.bin”) if !defined $name;”

#IMPORTANT

follow the next steps to make your terminal able to read the pintos command

1. open terminal and type:

cd ~

2. now you are at $HOME type:

nano .bashrc

3. nano will open, just add the following lane to the last line of code (add a newline)

“export PATH=$HOME/os-pintos/pintos/src/utils:$PATH”

4. restart the terminal

now head to ../utils/ using terminal and run any pintos command.