Introduction to Pintos

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Welcome to Pintos!

What is Pintos?

- An instructional operating system
- Developed by Ben Pfaff @ Stanford U.
- A real, bootable OS for 80x86 architecture
  - Run on a regular IBM-compatible PC or an x86 simulator
- The original structure and form was inspired by the Nachos instructional OS from UC Berkeley (Java-based)
- A few of the sources files are derived from code used in the MIT’s advanced operating systems course
- Written in C language (with minimal assembly code)
Bochs (1)

- What is Bochs?
  - Open-source IA-32 emulator
  - Simulates a complete Intel x86 computer in software
    - Interprets every instruction from power-up to reboot
    - Has device models for all of the standard PC peripherals: keyboard, mouse, VGA card/monitor, disks, timer, network, ...
    - Supports many different host platforms: x86, PowerPC, Alpha, Sun, and MIPS
  - Runs most popular x86 OSes:
    - Windows 95/98/NT/2000/XP/Vista/7, Linux, BSDs, ...
  - Written in C++
  - Emulation, not virtualization
Bochs (2)

- Linux + Bochs
  - We will run Pintos using Bochs on Linux
  - Bochs makes it easy to develop and debug Pintos projects
Bootstrapping Pintos

- Step 1: Get Linux ready
- Step 2: Get Pintos ready
- Step 3: Get Bochs ready
- Step 4: Test if everything works fine
Step 1: Linux (1)

- Install a Linux distribution on your machine
  - Ubuntu, Fedora, Debian, ...

- Recommended platform for this semester
  - Ubuntu Desktop 13.04 LTS 32-bit version
  - Easier to get help from TA & instructor
  - Available from
    http://www.ubuntu.com/download/desktop
  - Installation manual:
    http://www.ubuntu.com/download/help/install-ubuntu-desktop
Step 1: Linux (2)

- **Installation on a native machine**
  - Allocate a dedicated partition for Linux
  - Can co-exist with Windows
  - Fast!

- **Installation on a virtual machine**
  - VMware Player (free) or VMware Workstation ($$$)
  - Available from [http://www.vmware.com](http://www.vmware.com)
  - Pre-installed VM images are available on the Internet
    - VMware virtual appliances: [https://solutionexchange.vmware.com/store/category_groups/19](https://solutionexchange.vmware.com/store/category_groups/19)
    - e.g., Ubuntu 12.04 LTS image: [http://www.trendsigma.net/vmware/ubuntu1204.htm](http://www.trendsigma.net/vmware/ubuntu1204.htm)
  - (cf.) [http://csl.skku.edu/CSE2003S09/Linux](http://csl.skku.edu/CSE2003S09/Linux)
Step 1: Linux (3)

- Install basic development tools

  $ sudo apt-get install <package>

- build-essential
- vim, ctags, cscope
- patch, diff
- wget
- perl
- subversion, git
- ...

SSE3044: Operating Systems | Fall 2013 | Jin-Soo Kim (jinsookim@skku.edu)
Step 2: Pintos

- Installing Pintos
  - Available from http://csl.skku.edu/SSE3044F13/Resources
    - Use this version only
  - Use the following command to download Pintos
    
    ```bash
    $ cd ~
    $ wget http://csl.skku.edu/uploads/SSE3044F13/pintos.tar.gz
    ```
  - Untar Pintos
    
    ```bash
    $ tar xvzf pintos.tar.gz
    ```
  - Now, Pintos source code is in ~/pintos/src
Step 3: Bochs (1)

- You need Bochs to run Pintos
  - Do not install the Bochs package from Ubuntu repositories
  - If you did, it should be removed

- Install prerequisites for building Bochs
  - Install X windows development libraries
    
    $ sudo apt-get install xorg-dev
  
  - Install curses development libraries
    
    $ sudo apt-get install libncurses5-dev
Step 3: Bochs (2)

- **Install pre-built GCC 4.1.2 for Bochs**
  - Bochs is not compiled with the default compiler in Ubuntu (gcc-4.6.3)
  - We have pre-built gcc/g++ version 4.1.2 for you
  - Use our installation script
    
    $ chmod a+x sse-gcc-installer-v3
    $ sudo ./sse-gcc-installer-v3
    
    - The script will fetch and untar sse-gcc-4.1.2-[32bit|64bit].tar.gz file in your system
      - /usr/local/bin/sse-gcc, /usr/local/bin/sse-g++
  - /usr/local/bin should be in your PATH
Step 3: Bochs (3)

- **Download Bochs**
  - From [http://bochs.sourceforge.net](http://bochs.sourceforge.net)
  - Make sure you are downloading bochs-2.2.6.tar.gz
  - You don’t have to untar the source code

- **Installing Bochs**
  - Must patch the Bochs source code for Pintos
    - Patches are available in ~/pintos/src/misc
  - Use the installation script provided by Pintos:
    - ~/pintos/src/misc/bochs-2.2.6-build.sh
    - This will untar, patch, configure, compile, and install Bochs
  - You need to be a superuser (root) to install Bochs in the system directory (e.g., /usr/local/bin)
Step 3: Bochs (4)

- Running the script

```bash
$ cd ~/pintos/src/misc
$ sudo env SRCDIR=/home/jinsoo PINTOSDIR=/home/jinsoo/pintos DSTDIR=/usr/local CC=sse-gcc CXX=sse-g++ ./bochs-2.2.6-build.sh
```

- **SRCDIR**: The directory where the bochs-2.2.6.tar.gz is located
- **PINTOSDIR**: The base directory of Pintos
- **DSTDIR**: The target directory where Bochs will be installed
- **CC=sse-gcc, CXX=sse-g++**: Use sse-gcc/sse-g++ for C/C++ compiler
Step 4: Testing (1)

- Testing Bochs
  
  $ bochs \quad ; \text{Put } $DSTDIR/bin\text{ into your PATH}$
Step 4: Testing (2)

Building Pintos

- Specify `sse-gcc` in `~/pintos/src/Make.config`
  - At line 14 & 19: `gcc → sse-gcc`
  - (cf.) [http://csl.skku.edu/SSE3044F12/GCCPintos](http://csl.skku.edu/SSE3044F12/GCCPintos)

```
$ cd ~/pintos/src/threads
$ make
```

- This will create the kernel image (`kernel.bin`) and the final OS disk image (`os.dsk`) in `~/pintos/src/threads/build`
- `os.dsk = loader.bin + kernel.bin`
Step 4: Testing (3)

- **Run Pintos**

  $ cd pintos/src/threads

  $ ../utils/pintos run alarm-multiple

```
(alarm-multiple) thread 0: duration=10, iteration=7, product=70
(alarm-multiple) thread 1: duration=20, iteration=4, product=80
(alarm-multiple) thread 3: duration=40, iteration=2, product=80
(alarm-multiple) thread 2: duration=30, iteration=3, product=90
(alarm-multiple) thread 4: duration=50, iteration=2, product=100
(alarm-multiple) thread 1: duration=20, iteration=5, product=100
(alarm-multiple) thread 2: duration=30, iteration=4, product=120
(alarm-multiple) thread 3: duration=40, iteration=3, product=120
(alarm-multiple) thread 1: duration=20, iteration=6, product=120
(alarm-multiple) thread 2: duration=30, iteration=5, product=140
(alarm-multiple) thread 3: duration=40, iteration=5, product=150
(alarm-multiple) thread 4: duration=50, iteration=7, product=150
(alarm-multiple) thread 2: duration=30, iteration=4, product=160
(alarm-multiple) thread 3: duration=40, iteration=6, product=160
(alarm-multiple) thread 4: duration=50, iteration=4, product=200
(alarm-multiple) thread 3: duration=40, iteration=5, product=200
(alarm-multiple) thread 2: duration=30, iteration=7, product=210
(alarm-multiple) thread 3: duration=40, iteration=6, product=240
(alarm-multiple) thread 4: duration=50, iteration=5, product=250
(alarm-multiple) thread 3: duration=40, iteration=7, product=280
(alarm-multiple) thread 4: duration=50, iteration=6, product=300
(alarm-multiple) thread 4: duration=50, iteration=7, product=350
(alarm-multiple) end
Execution of 'alarm-multiple' complete.
```
Project -1: Warming Up
Project -1 (1)

- Set up your own project environment
  - Install Linux
  - Install all the required tools
  - Install Pintos
  - Install Bochs
  - Capture the screen shot of working Pintos
    $ pintos run alarm-multiple
Project -1 (2)

- Documentation
  - Specification of your environment
    - Hardware info
    - Linux distributions
    - Output of the following commands:
      - `$ uname -a`  
      - `$ lscpu`  
    - A screen shot of “alarm-multiple”

- Due:
  - Sep. 6 (Fri), 11:59PM (firm deadline)
  - Submit via e-mail to sse3044@cs1.skku.edu
  - Note: This is an individual project