Teaching Assistants

- Jaemin Lee
  - E-mail: jminlee92 at gmail.com
  - Office: #85533

- Inyoung Park
  - E-mail: kisys18 at gmail.com
  - Office: #85557

- E-mail is the preferred way to contact
- Make an appointment before you visit
Overview

- Setting up your Pintos
- Project schedule & policies
- Project 0: Warming up
Welcome to Pintos!

- **Review: 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
  - Written in C language with minimal assembly code
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, Linux, BSDs, ...
- Written in C++
- Emulation, not virtualization
Linux + Bochs

- We can run Pintos using Bochs on Linux
- Bochs makes it easy to develop and debug Pintos projects
Setting Up with Bochs (1)

- Prerequisite of Pintos + Bochs
  - Install Ubuntu on your machine
  - `sudo apt-get install`
    - `patch`
    - `diff`
    - `g++`
    - `xorg-dev`
    - `ncurses-dev`
  - `sudo apt-get update`
  - There could be additional libraries/tools to install
Setting Up with Bochs (2)

- Install Pintos
  - Download the Pintos package (pintos.tar.gz)
    - Use this version only
  - Untar Pintos
    $ tar xvzf pintos.tar.gz
  - Build Pintos
    $ cd pintos/src/threads
    $ make
    - This will create the kernel image (kernel.bin) and the final OS disk image (os.dsk = loader.bin + kernel.bin) in ./build
Setting Up with Bochs (3)

- Install Bochs
  - Get the source code from [http://bochs.sourceforge.net](http://bochs.sourceforge.net)
    - Make sure you are downloading v2.2.6 (bochs-2.2.6.tar.gz)
    - You don’t have to untar the source code
  - Install 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)
    - The script 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)
Setting Up with Bochs (4)

- Install Bochs (cont’d)
  - Running the script:

```
$ sudo env SRCDIR=/home/jminlee/ PINTOSDIR=/home/jminlee/OS_project/pintos/ DSTDIR=/usr/local/ ./bochs-2.2.6-build.sh
```
Setting Up with Bochs (5)

- Test Bochs
  
  $ bochs ; Put $DSTDIR/bin into your PATH
Setting Up with Bochs (6)

- Setting pintos-gdb

```bash
$ vi pintos/src/utils/pintos-gdb
```

```bash
#!/bin/sh

# Path to GDB macros file. Customize for your site.
GDBMACROS=/home/jminlee/OS_project/pintos/src/misc/gdb-macros

# Choose correct GDB.
if command -v i386-elf-gdb >/dev/null 2>&1; then
    GDB=i386-elf-gdb
else
    GDB=gdb
fi

# Run GDB.
if test -f "$GDBMACROS"; then
    exec $GDB -x "$GDBMACROS" "$@
else
    echo "*** $GDBMACROS does not exist ***"
    echo "*** Pintos GDB macros will not be available ***"
    exec $GDB "$@
fi
```
Setting Up with Bochs (7)

- Run Pintos

```bash
$ 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 2: duration=30, iteration=3, product=90
(alarm-multiple) thread 3: duration=40, iteration=2, product=80
(alarm-multiple) thread 4: duration=50, iteration=1, 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=7, product=140
(alarm-multiple) thread 4: duration=50, iteration=6, product=150
(alarm-multiple) thread 3: duration=40, iteration=8, product=160
(alarm-multiple) thread 2: duration=30, iteration=9, product=160
(alarm-multiple) thread 4: duration=50, iteration=7, product=200
(alarm-multiple) thread 3: duration=40, iteration=8, product=200
(alarm-multiple) thread 2: duration=30, iteration=9, product=210
(alarm-multiple) thread 4: duration=50, iteration=7, product=240
(alarm-multiple) thread 3: duration=40, iteration=8, product=250
(alarm-multiple) thread 4: duration=50, iteration=7, product=300
(alarm-multiple) thread 4: duration=50, iteration=7, product=350
(alarm-multiple) end
Execution of 'alarm-multiple' complete.
```
Qemu

- What is Qemu?
  - Quick EMUlator
  - Written by Fabrice Bellard
  - Supports the emulation of various architectures
    - IA-32, x86-64, MIPS R4000, Sun, ARM, PowerPC, etc..

- Qemu + Linux
  - We can run Pintos using Qemu on Linux
  - Installation of Qemu is very easy!
Setting Up with Qemu (1)

- Install Ubuntu on your machine
- Install QEMU
  - See [http://csl.skku.edu/SSE3044F12/QEMU](http://csl.skku.edu/SSE3044F12/QEMU)
Setting Up with Qemu (2)

- **Install Pintos**
  - Download the Pintos package (pintos.tar.gz)
    - Use this version only
  - Untar Pintos
    $ tar xvzf pintos.tar.gz
  - Build Pintos
    $ cd pintos/src/threads
    $ make
    - This will create the kernel image (kernel.bin) and the final OS disk image (os.dsk = loader.bin + kernel.bin) in ./build
Setting Up with Qemu (3)

- Setting Pintos for QEMU
  - Simulator Setting
    - Check Make.vars at ~/pintos/src/threads
    - ‘Simulator = --qemu’
  - Pintos script setting
    - Also see http://csl.skku.edu/SSE3044F12/QEMU
    - Modify ~/pintos/src/utils/pintos
    - You can use any text editor to modify this
  - Run option
    - You have to use --qemu option for pintos
      - Default simulator is bochs
    - ../utils/pintos --qemu -- run alarm-multiple
What is different?

- **Difference between Bochs and Qemu**
  - “Reproducibility” is important issue for debugging
    - Always same result occurs when you run program in same manner
  - Bochs offers reproducibility
    - Same jitter value causes exactly same result
    - But it also provides real time mode
      - By using –r option
  - Qemu doesn’t offer reproducibility
    - Only real time mode is supported
    - Qemu is faster
Projects

- Project 0: Warming up
  - pintos/src/threads
- Project 1: Threads
  - pintos/src/threads
- Project 2: User programs
  - pintos/src/userprog
- Project 3: Virtual memory
  - pintos/src/vm
- Project 4: File system
  - pintos/src/filesys

Use “make” command in each of project directories
Interesting files in the ./build directory

- **kernel.o:**
  - The object file for the entire kernel
  - Used for debugging

- **kernel.bin:**
  - The memory image of the kernel

- **loader.bin:**
  - The memory image of the kernel loader (512 bytes)
  - Reads the kernel from disk into memory and starts it up

- **os.dsk:**
  - Disk image for the kernel (loader.bin + kernel.bin)
  - Used as a “virtual disk” by the simulator
A Tour of Pintos (3)

- **Running Pintos**
  - Add “pintos/src/utils” to $PATH and run “pintos”
    - $ export PATH=“~/pintos/src/utils:$PATH”
    - $ pintos [option] -- [argument]
  - **Option**
    - Configure the simulator or the virtual hardware
  - **Argument**
    - Each argument is passed to the Pintos kernel verbatim
    - ‘pintos run alarm-multiple’ instructs the kernel to run alarm-multiple
  - **Pintos script**
    - Parse command line, find disks, prepare arguments, run the simulator (Bochs)
A Tour of Pintos (4)

- Project testing
  
  $ make check

  $ make grade

```
FAIL tests/threads/alarm-single
FAIL tests/threads/alarm-multiple
pass tests/threads/alarm-simultaneous
FAIL tests/threads/alarm-priority
pass tests/threads/alarm-zero
pass tests/threads/alarm-negative
FAIL tests/threads/priority-change
FAIL tests/threads/priority-donate-one
FAIL tests/threads/priority-donate-multiple
FAIL tests/threads/priority-donate-multiplex
FAIL tests/threads/priority-donate-nest
FAIL tests/threads/priority-donate-same
FAIL tests/threads/priority-donate-lower
FAIL tests/threads/priority-lifo
FAIL tests/threads/priority-preempt
FAIL tests/threads/priority-sema
FAIL tests/threads/priority-condvar
FAIL tests/threads/priority-donate-chain
FAIL tests/threads/mfqs-load-1
FAIL tests/threads/mfqs-load-60
FAIL tests/threads/mfqs-load-avg
FAIL tests/threads/mfqs-recent-1
pass tests/threads/mfqs-Fair-2
pass tests/threads/mfqs-Fair-20
FAIL tests/threads/mfqs-nice-2
FAIL tests/threads/mfqs-nice-10
FAIL tests/threads/mfqs-block
22 of 27 tests failed.
makes: [check] Error 1
```

$
A Tour of Pintos (5)

- Useful tools
  - gdb: The GNU project debugger
    - Allows to see what’s going on inside another program while it executes
    - Refer to Appendix E.5: GDB
  - Tags
    - An index to the functions and global variables
    - Powerful when it is combined with vi editor
    - Refer to Appendix F.1: Tags
  - CVS: Version-control system
    - Useful for version controls and concurrent development
    - Refer to Appendix F.3: CVS
    - You can also use SVN or Git
A Tour of Pintos (6)

- **Tips**
  - Read the project specification carefully
  - Before starting your project, read the document template too!
    - It may give you useful tips
  - Study the test cases in `pintos/src/tests` used by “make check”
    - One C program for each test case (*.c)
    - One Perl script to check whether your implementation is correct or not (*.ck)
    - Study the correct output stored in the perl script
  - Do it incrementally
    - Otherwise, it can be totally messed up
Project Schedule

- **Project 0**
  - Warming up project  (3/17~3/23)

- **Project 1**
  - Threads  (3/31~4/13)

- **Project 2**
  - User programs  (4/28~5/18)

- **Project 3**
  - Virtual memory  (5/19~6/10)

- Project 1, 2 and 3 are team project (2 students)
- This schedule is subject to change
Project Policy (1)

- Cheating policy
  - “Copying all or part of another person’s work, or using reference material not specifically allowed, are forms of cheating and will not be tolerated.”
  - For a student involved in an incident of cheating, the following policy will apply:
    - You will get a penalty in the final grade (down to F)
    - For serious offenses, this will be notified to the department chair
  - Share useful information: helping others use systems or tools, helping them with high-level designs or debug their code is NOT cheating!
  - To check cheating, TA see submissions, analyze codes & ask
Project Policy (2)

- Late policy
  - 20% off per day after due date
Project Grading (1)

- **Functionality (70%)**
  - `$ make check`
  - `$ make grade`

- **Design & documentation (30%)**
  - **Source code**
    - variable name, function name, comments
  - **Design document**
    - Data structure, Algorithm, Synchronization, Rationale
  - Refer to Appendix D: Project Documentation

- **Demos & oral tests**
Project Grading (2)

- Source code
- Comments

```
NTSTATUS
FatCommonCreate {
    __inout PIRP_CONTEXT IrpContext,
    __inout PIRP Irp
}

/**
Routine Description:

  This is the common routine for creating/opening a file called by
  both the fsd and fsp threads.

Arguments:

  Irp - Supplies the Irp to process

Return Value:

  NTSTATUS - the return status for the operation
*/

DebugTrace(0, Dbg, "->Ealength = %08lx\n", IrpSp->Parameters
    .Create.Ealength);

// This is here because the Win32 layer can't avoid sending me double
// beginning backslashes.
//
if ((IrSp->FileObject->FileName.Length > sizeof(WCHAR)) &&
    (IrSp->FileObject->FileName.Buffer[0] == L'\') &&
    (IrSp->FileObject->FileName.Buffer[1] == L'\')) {

    IrpSp->FileObject->FileName.Length -= sizeof(WCHAR);

    RtlMoveMemory( &IrSp->FileObject->FileName.Buffer[0],
        &IrSp->FileObject->FileName.Buffer[1],
        IrpSp->FileObject->FileName.Length );

    //
    // If there are still two beginning backslashes, the name is bogus.
    //
    if ((IrSp->FileObject->FileName.Length > sizeof(WCHAR)) &&
```
Demos & oral tests

- Usually done in the next week of the due date
- Everyone should meet the TA in the office
- You may bring your notebook as there could be a problem in running your solution in the TA's machine
- You should be able to answer any questions on
  - Basic system architecture
  - Design decisions
  - Implementation details
  - ...
PROJECT 0: WARMING UP
Project 0 (1)

- Set up your own project environment
  - Install Linux
  - Install all the required tools
  - Install Pintos
Project 0 (2)

- Add a new test code: print-name
  - Add a new kernel function which prints your name in ASCII text format
  - To run the new function, add a new command “print-name”
    - The following command should run your new function
      $ pintos run print-name
  - Work in the pintos/src/threads and pintos/src/tests/threads directories
Add a new test code: print-name

- Print format
  - (print-name) Course : SWE3004
  - (print-name) ID     : 20100000000
  - (print-name) Name   : Jaemin Lee
Example:

Bochs x86 emulator, http://bochs.sourceforge.net/

Bochs BIOS - build: 01/25/06
$Revision: 1.160 $ $Date: 2006/01/25 17:51:49 $
Options: apmbios pcibios eltorito

ata0 master: Generic 1234 ATA-2 Hard-Disk (0 MBytes)

Booting from Hard Disk...
PILo hda1
Loading.........
Kernel command line: run print-name
Pintos booting with 4,096 kB RAM...
383 pages available in kernel pool.
383 pages available in user pool.
Calibrating timer... 204,600 loops/s.
Boot complete.
Executing 'print-name':
(print-name) begin
(print-name) Course: SSE3004
(print-name) ID: 2010000000
(print-name) Name: Jaemin Lee
(print-name) end
Execution of 'print-name' complete.
Submission (1)

- **Documentation**
  - A screen shot of “alarm-multiple”
  - A screen shot of “print-name”
  - Detailed explanation of how the “print-name” is handled and your name is printed by the kernel
  - File format – PDF format
  - File name – “GDHong_2013123456.pdf”

- **Source code**
  - Tar and gzip your Pintos source codes
    - `$ cd pintos`
    - `$ (cd src/threads; make clean)`
    - `$ tar cvzf GDHong_2013123456.tar.gz src`
Submission (2)

- Due
  - Mar. 23, 11:59PM
  - Upload your source code and documentation at iCampus
- Good luck!