Introduction to Pintos

Prof. Jin-Soo Kim (jinsookim@skku.edu)
TAs – Jong-Sung Lee (leitia@csl.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

A Tour of Pintos (1)

- **Projects**
  - 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

A Tour of Pintos (2)

- **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
```

```
xterm
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-multiple2
FAIL tests/threads/priority-donate-nest
FAIL tests/threads/priority-donate-sema
FAIL tests/threads/priority-donate-lower
FAIL tests/threads/priority-fifo
FAIL tests/threads/priority-preempt
FAIL tests/threads/priority-sema
FAIL tests/threads/priority-condvar
FAIL tests/threads/priority-donate-chain
FAIL tests/threads/mlfqs-load-1
FAIL tests/threads/mlfqs-load-60
FAIL tests/threads/mlfqs-load-avg
FAIL tests/threads/mlfqs-recent-1
pass tests/threads/mlfqs-fair-2
pass tests/threads/mlfqs-fair-20
FAIL tests/threads/mlfqs-nice-2
FAIL tests/threads/mlfqs-nice-10
FAIL tests/threads/mlfqs-block
22 of 27 tests failed.
make: *** [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

A Tour of Pintos (6)

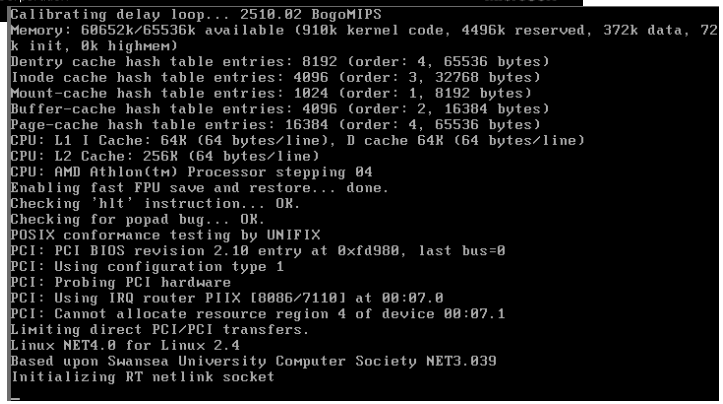
- **Tips**
  - Read the project specification carefully
  - Before starting your project, read the document template too!
    - It may give you useful tips
  - Study the test cases in `pintos/src/tests` used by "make check"
    - One C program for each test case (*.c)
    - One Perl script to check whether 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

System Startup

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

System Startup (1)

- **Overview**
  - BIOS
  - Boot loader
  - Kernel initialization



System Startup (2)

- **The BIOS**
  - The CPU initializes itself and then begins to execute an instruction at a fixed location (`0xffff fff0`)
  - Those instructions are supplied from ROM and make the CPU jump into the BIOS
  - The BIOS finds a boot device and loads its first sector into memory
    - Starting from physical address `0x0000 7c00`
    - The first sector contains the Pintos' loader (`threads/loader.S`)
  - The BIOS transfers control to the loader

System Startup (3)

- **The boot loader**
  - Enables memory accesses beyond first 1MB
    - For historical reasons, this initialization is required
  - Asks the BIOS for the PC's memory size
    - Again for historical reasons, the function we use can only detect up to 64MB of RAM (This is the limit that Pintos can support)
    - The memory size is stored in the loader and the kernel can read the information after it boots
  - Creates a basic page table
    - This page table maps the 64MB at the base (starting at virtual address 0) directly to identical physical address
    - It also maps the same physical memory starting at virtual address LOADER_PHYS_BASE (`0xc000 0000`)

System Startup (4)

- **The boot loader (cont'd)**
  - Turns on protected mode and paging
    - Interrupts are still disabled
  - Loads the kernel from disk
    - Assumptions:
      - » The kernel is stored starting from the second sector of the first IDE disk
      - » The BIOS has already set up the IDE controller
    - The loader loads the kernel starting at physical address LOADER_KERN_BASE (`0x0010 0000`)
  - Jumps to the kernel entry point
    - main() in src/threads/init.c
    - Set up using the linker script (`threads/kernel.lds.S`)

System Startup (5)

- **Kernel initialization**
  - Clears BSS and get machine's RAM size
  - Initializes threads system
  - Initializes VGA, serial port, and console
    - To print a startup message to the console
  - Greets user and reading kernel command line
    - "Kernel command line: "
  - Initializes memory system
  - Initializes random number generator and interrupt system
  - Starts thread scheduler and enables interrupts
  - Initializes file system

Project Policies

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

Project Schedule

- **Project 0**
  - Warming-up project (1 weeks, 9/9~9/15)
- **Project 1**
  - Threads (2 weeks, 9/16~9/29)
- **Project 2**
  - User programs (4 weeks, 10/7~11/3)
- **Project 3**
  - Virtual memory (4 weeks, 11/11~12/8)
- **This schedule is subject to change**

Project Policy (1)

- **Late policy**
  - 30% off per day after due date.

Project Policy (2)

- **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 submission server, analyze detail code & ask

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

```
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
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

--*/

                                                    377,0-1        5%
```

```
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
    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 ((IrpSp->FileObject->FileName.Length > sizeof(WCHAR)) &&
        (IrpSp->FileObject->FileName.Buffer[1] == L'\\') &&
        (IrpSp->FileObject->FileName.Buffer[0] == L'\\')) {

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

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

        //
        //  If there are still two beginning backslashes, the name is bogus.
        //

        if ((IrpSp->FileObject->FileName.Length > sizeof(WCHAR)) &&
                                                    479,0-1        7%
```

Project Grading (3)

- **Demos & oral tests**
  - Usually done in the next week of the due date
  - Everyone should meet the TA offline
  - 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

SKK UNIVERSITY

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

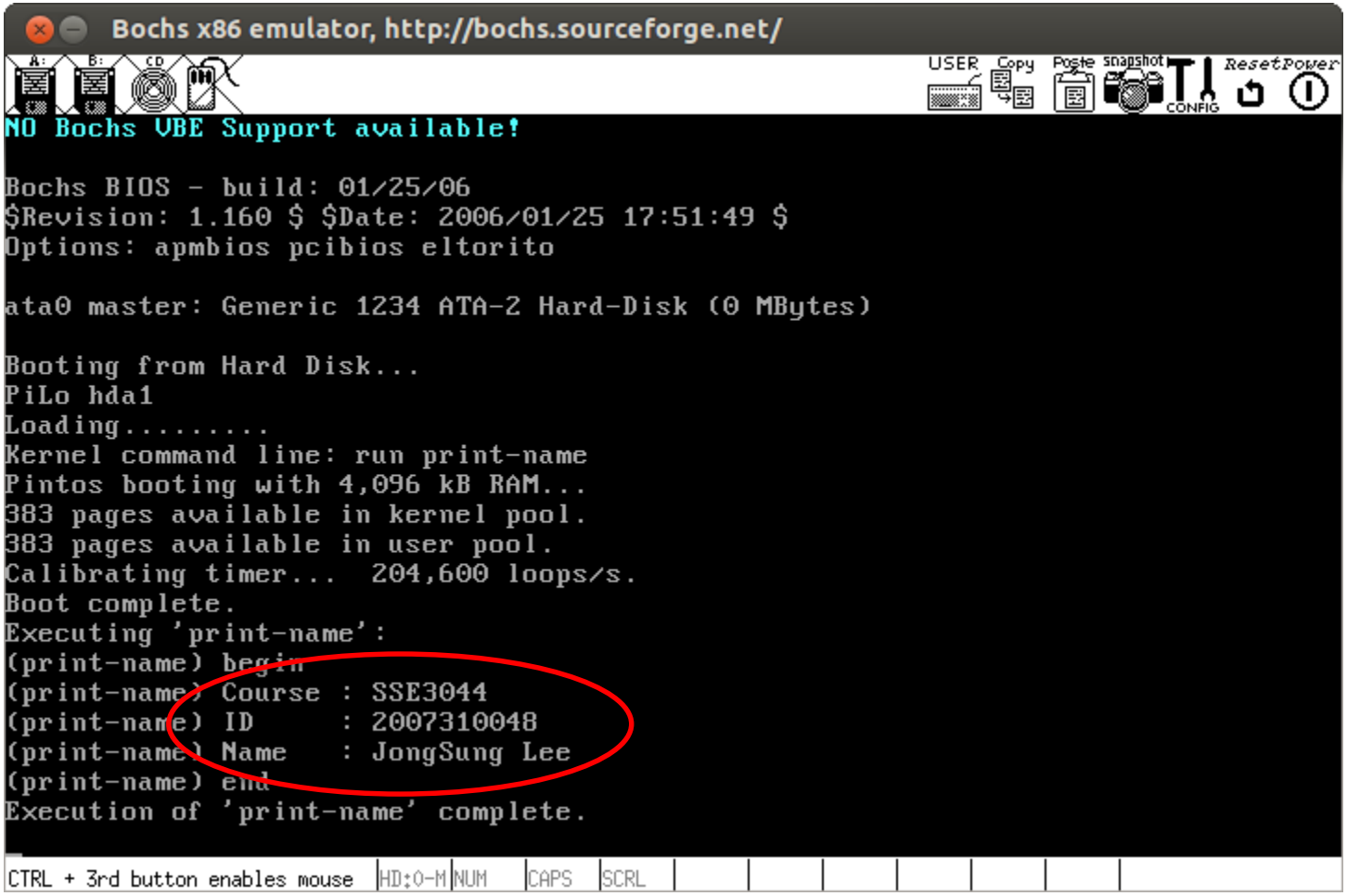
      ```
      $ pintos run print-name
      ```
  - Work in the `pintos/src/threads` and `pintos/src/tests/threads` directories

Project 0 (3)

- **Add a new test code: print-name**
  - Print format
    - (print-name) Course : SSE3044
    - (print-name) ID : 2010000000
    - (print-name) Name : GilDong Hong
  - Capture screenshot

Project 0 (4)

- **Example:**



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**
  - Sep. 15, 11:59PM
  - Submit your source code and documentation via sse3044@csl.skku.edu
  - Good luck!