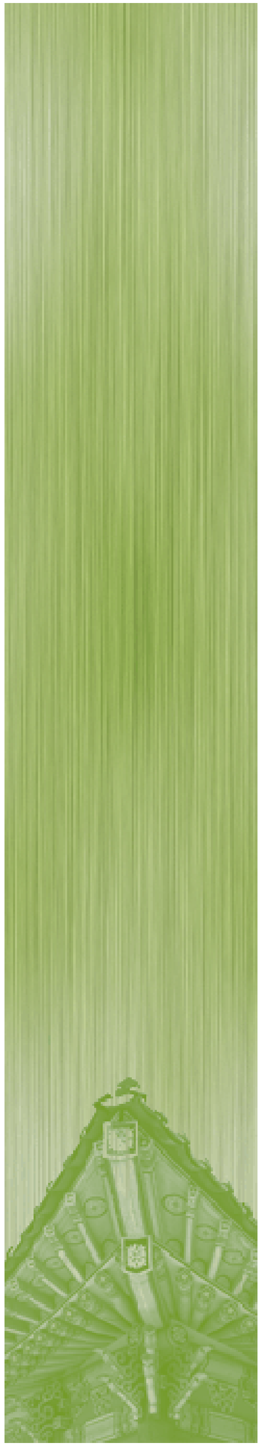


Introduction to Pintos

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?

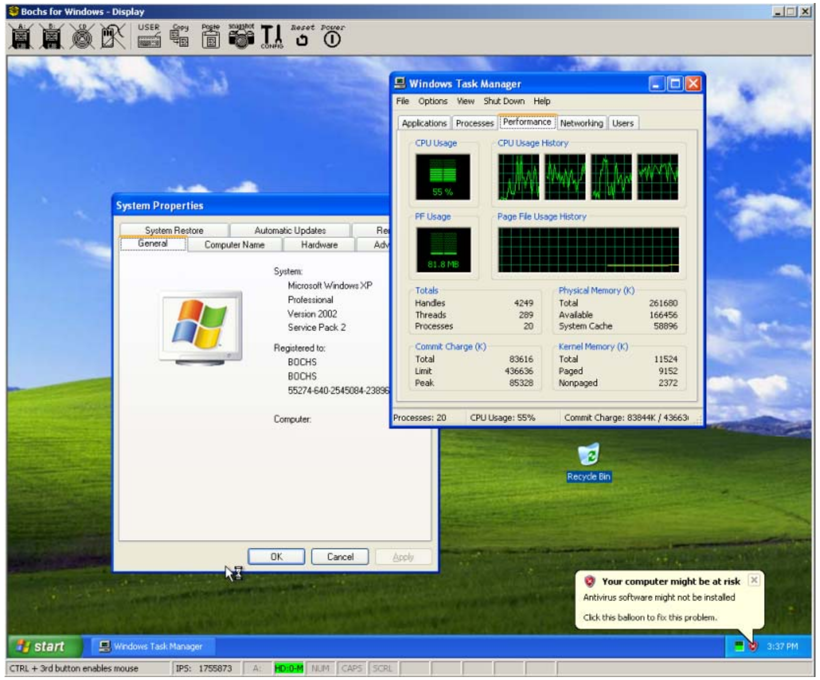
- Open-source IA-32 emulator
- Simulates a complete Intel x86 computer in software
  - Interprets every instruction from power-up to reboot
  - Has device models for all of the standard PC peripherals: keyboard, mouse, VGA card/monitor, disks, timer, network, …
  - Supports many different host platforms:
x86, PowerPC, Alpha, Sun, and MIPS
- Runs most popular x86 Oses:
  - Windows 95/98/NT/2000/XP/Vista, 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

barfy.bochs.com: VGA: Bochs
DOOM
id SOFTWARE
PROVIDED BY id FREE OF CHARGE ■ SUGGESTED RETAIL PRICE $9.00 ■ id SOFTWARE, ©1993


Bochs for Windows - Display
A: B: CD USER Copy Paste snapshot CONFIG Reset Power
Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
CPU Usage
55 %
CPU Usage History
PF Usage
81.8 MB
Page File Usage History
System Properties
System Restore Automatic Updates Re
General Computer Name Hardware Adv
OK Cancel Apply
Recycle Bin
start
Windows Task Manager
3:37 PM

System:
Microsoft Windows XP
Professional
Version 2002
Service Pack 2

Registered to:
BOCHS
BOCHS
55274-640-2545084-23896

Computer:

Totals	
Handles	4249
Threads	289
Processes	20

Physical Memory (K)	
Total	261680
Available	166456
System Cache	58896

Commit Charge (K)	
Total	83616
Limit	436636
Peak	85328

Kernel Memory (K)	
Total	11524
Paged	9152
Nonpaged	2372

Processes: 20 | CPU Usage: 55% | Commit Charge: 83844K / 43663

Your computer might be at risk
Antivirus software might not be installed
Click this balloon to fix this problem.

CTRL + 3rd button enables mouse | IPS: 1755873 | A: | HD:0-M | NUM | CAPS | SCRL

Setting Up (1)

- **Install Linux distribution on your machine**
  - Debian, Fedora, Ubuntu, or whatever you like
- **Install development tools**
  - Including gcc, make, perl, gdb, and so on
  - GCC >= 4.0, binutils >= 2.13
- **Install development libraries, (for Bochs)**
  - Install X windows development libraries, if needed
    - For Debian, install xorg-dev package
  - Install curses development libraries, if needed
    - For Debian, install libncurses5-dev package
  - There could be additional libraries to install

Setting Up (2)

- **Install Pintos**
  - Download the Pintos package (pintos.tar.gz)
    - Available from http://csl.skku.edu/uploads/CSE3008F09/pintos.tar.gz
    - Use this version only
  - Untar Pintos

```
$ tar xvzf pintos.tar.gz
```

  - Build Pintos

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

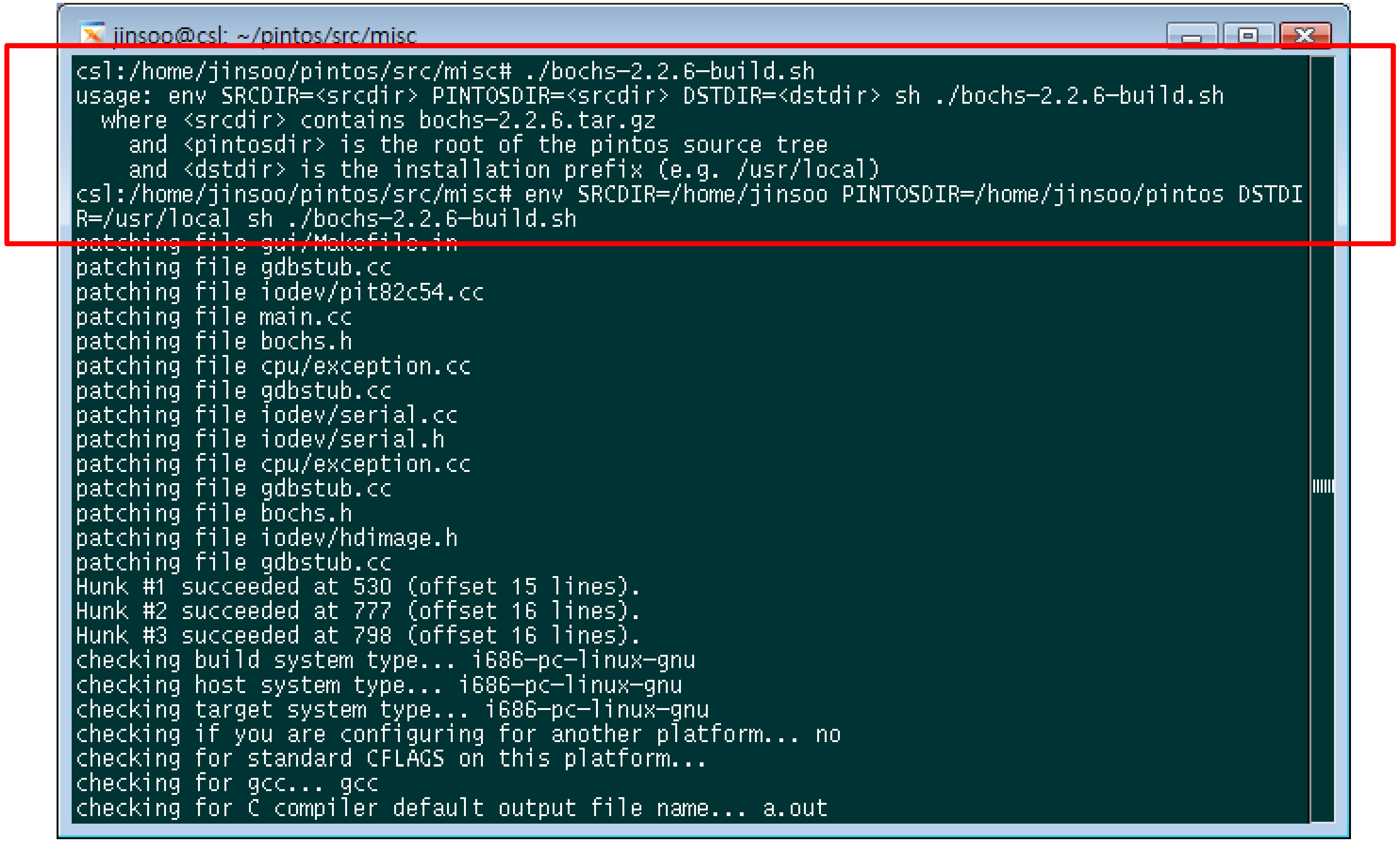
    - This will create the kernel image (kernel.bin) and the final OS disk image (os.dsk = loader.bin + kernel.bin) in ./build

Setting Up (3)

- **Install Bochs**
  - You need Bochs to run Pintos
  - Get the source code from 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 (4)

- **Install Bochs (cont'd)**
  - Running the script:

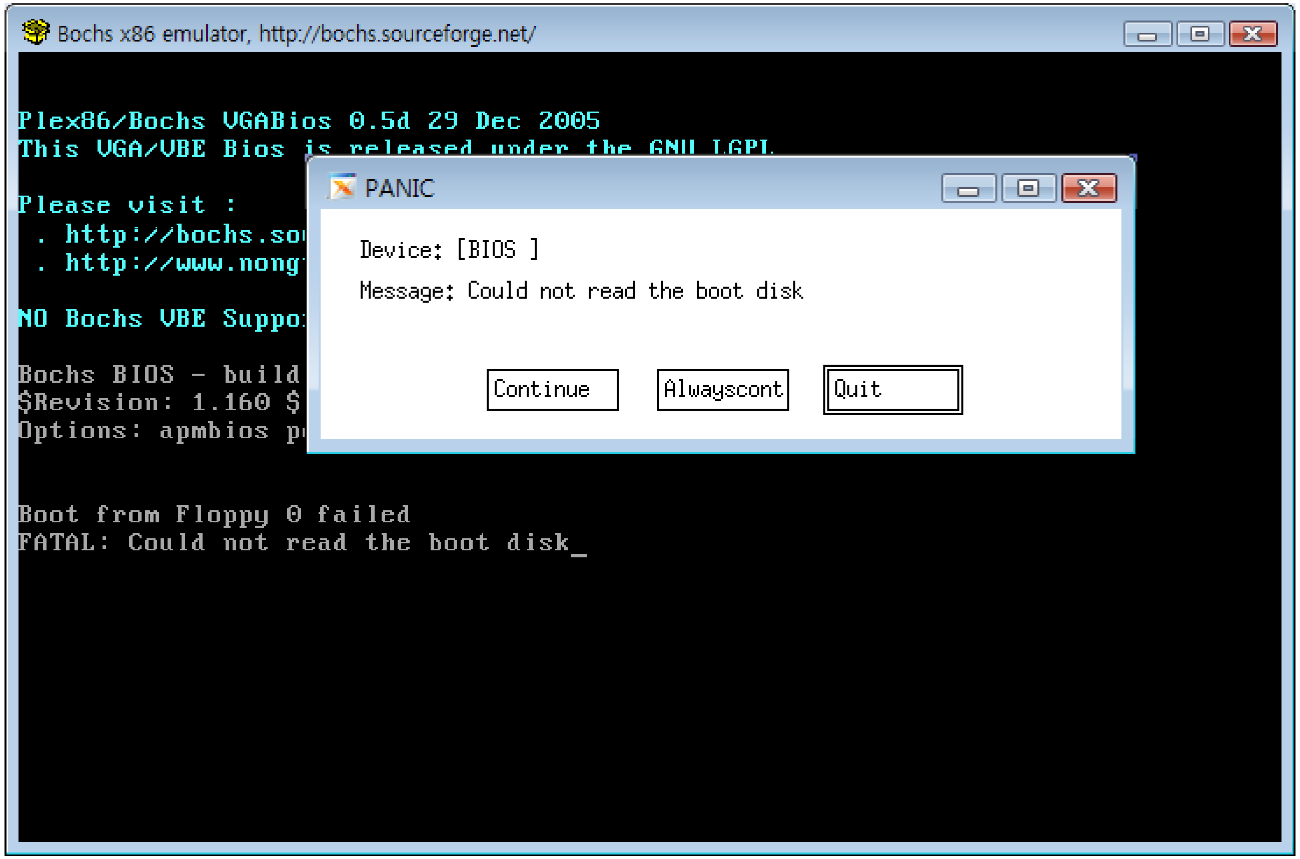


```
csl:/home/jinsoo/pintos/src/misc# ./bochs-2.2.6-build.sh
usage: env SRCDIR=<srcdir> PINTOSDIR=<srcdir> DSTDIR=<dstdir> sh ./bochs-2.2.6-build.sh
  where <srcdir> contains bochs-2.2.6.tar.gz
    and <pintosdir> is the root of the pintos source tree
    and <dstdir> is the installation prefix (e.g. /usr/local)
csl:/home/jinsoo/pintos/src/misc# env SRCDIR=/home/jinsoo PINTOSDIR=/home/jinsoo/pintos DSTDI
R=/usr/local sh ./bochs-2.2.6-build.sh
patching file gui/Makefile.in
patching file gdbstub.cc
patching file iodev/pit82c54.cc
patching file main.cc
patching file bochs.h
patching file cpu/exception.cc
patching file gdbstub.cc
patching file iodev/serial.cc
patching file iodev/serial.h
patching file cpu/exception.cc
patching file gdbstub.cc
patching file bochs.h
patching file iodev/hdimage.h
patching file gdbstub.cc
Hunk #1 succeeded at 530 (offset 15 lines).
Hunk #2 succeeded at 777 (offset 16 lines).
Hunk #3 succeeded at 798 (offset 16 lines).
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking if you are configuring for another platform... no
checking for standard CFLAGS on this platform...
checking for gcc... gcc
checking for C compiler default output file name... a.out
```

Setting Up (5)

- **Test Bochs**

```
$ bochs            ; Put $DSTDIR/bin into your PATH
```



```
Plex86/Bochs VGABios 0.5d 29 Dec 2005
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.so
 . http://www.nong

NO Bochs VBE Suppo

Bochs BIOS - build
$Revision: 1.160 $
Options: apmbios p


Boot from Floppy 0 failed
FATAL: Could not read the boot disk_
```

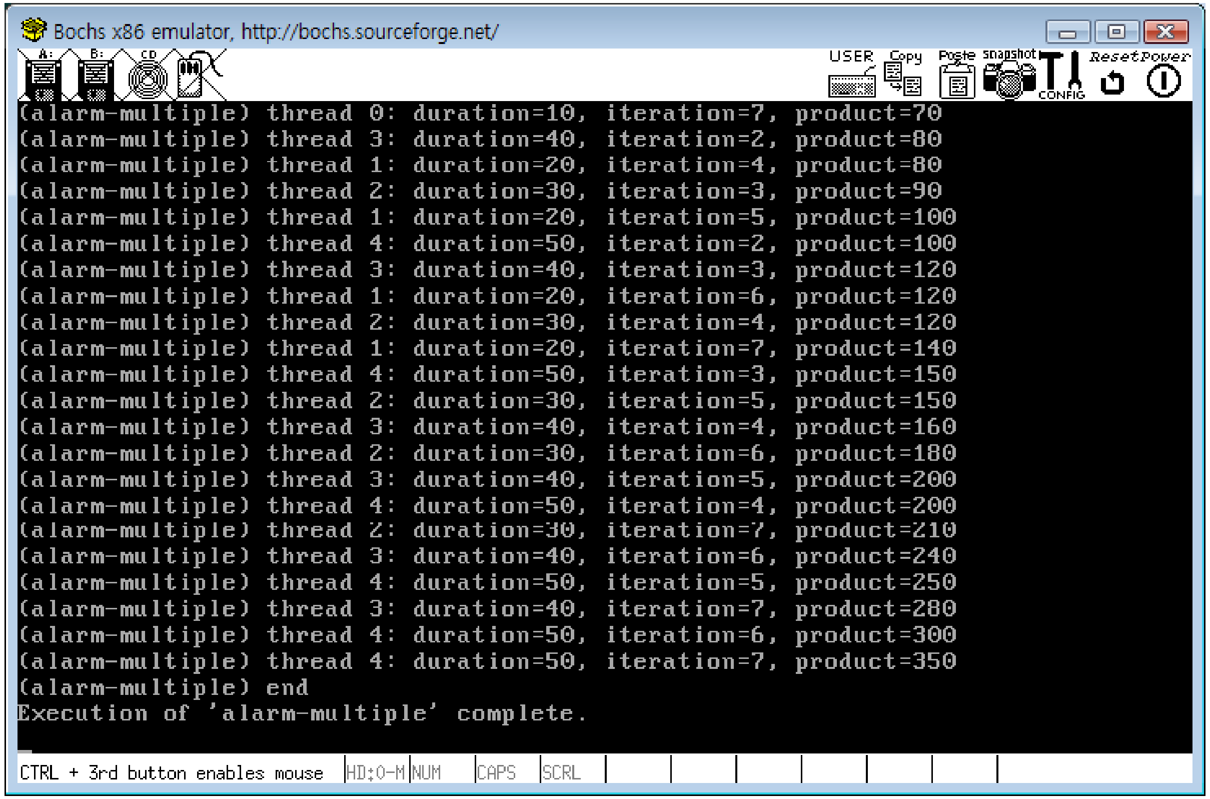
Device: [BIOS]

Message: Could not read the boot disk

Setting Up (6)

- **Run Pintos**

```
$ cd pintos/src/threads
$ ../utils/pintos run alarm-multiple
```



Bochs x86 emulator, http://bochs.sourceforge.net/
A:
B:
CD
USER
Copy
Paste
snapshot
CONFIG
Reset
Power
CTRL + 3rd button enables mouse
HD:0-M
NUM
CAPS
SCRL

```
(alarm-multiple) thread 0: duration=10, iteration=7, product=70
(alarm-multiple) thread 3: duration=40, iteration=2, product=80
(alarm-multiple) thread 1: duration=20, iteration=4, product=80
(alarm-multiple) thread 2: duration=30, iteration=3, product=90
(alarm-multiple) thread 1: duration=20, iteration=5, product=100
(alarm-multiple) thread 4: duration=50, iteration=2, product=100
(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=4, product=120
(alarm-multiple) thread 1: duration=20, iteration=7, product=140
(alarm-multiple) thread 4: duration=50, iteration=3, product=150
(alarm-multiple) thread 2: duration=30, iteration=5, product=150
(alarm-multiple) thread 3: duration=40, iteration=4, product=160
(alarm-multiple) thread 2: duration=30, iteration=6, product=180
(alarm-multiple) thread 3: duration=40, iteration=5, product=200
(alarm-multiple) thread 4: duration=50, iteration=4, 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.
```

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”

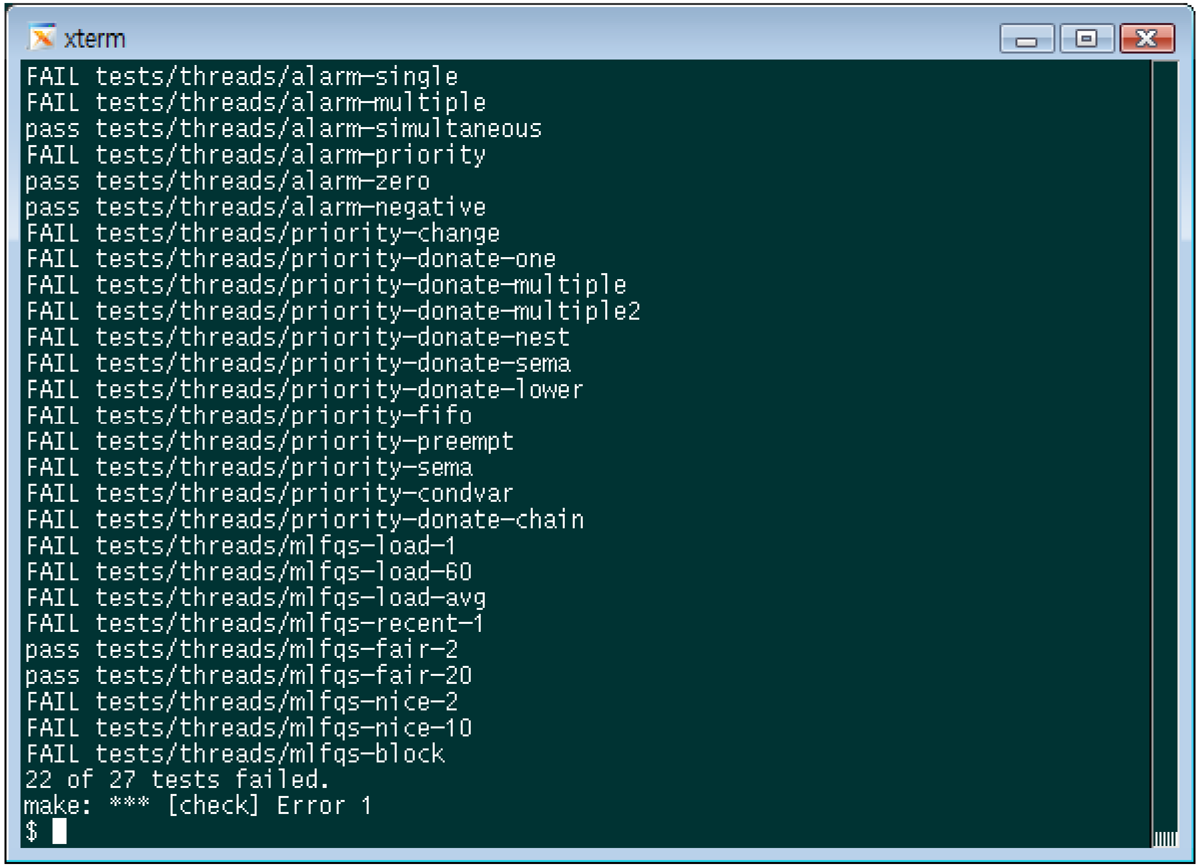
```
$ export PATH=“/home/jinsoo/pintos/src/utils:$PATH”
$ pintos [option] -- [argument]
```

  - 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-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
$ █
```

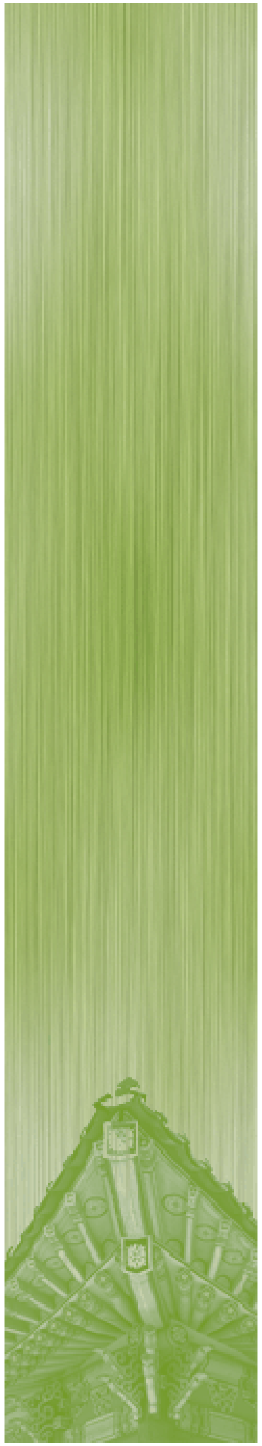
A Tour of Pintos (5)

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

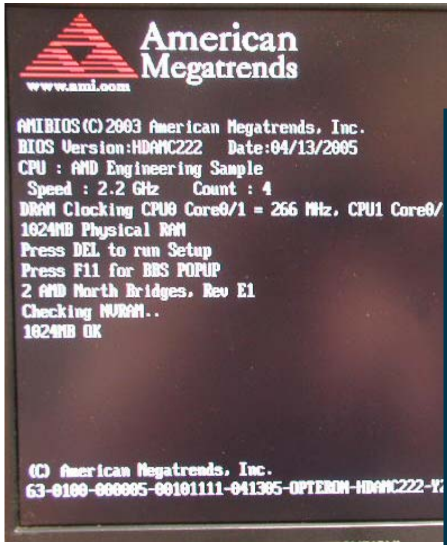


System Startup

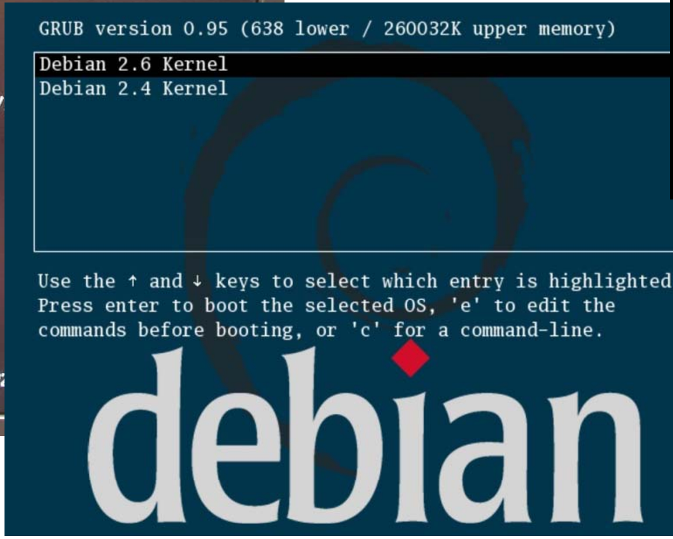
Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

System Startup (1)

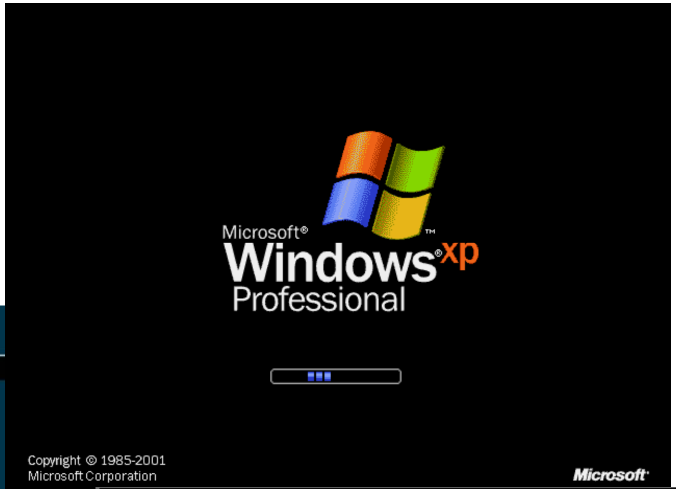
- **Overview**
  - BIOS
  - Boot loader
  - Kernel initialization



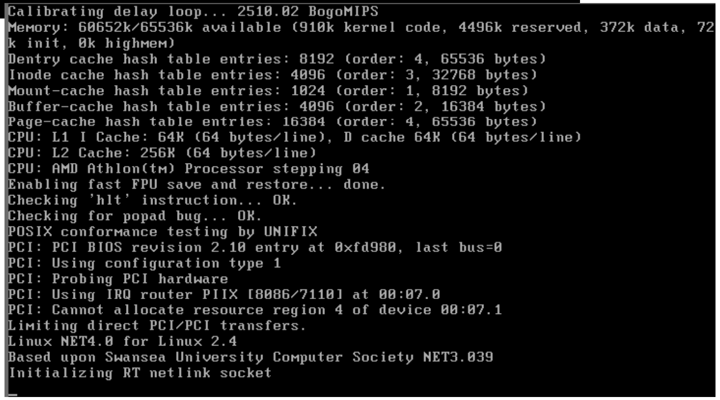
```
AMIBIOS(C)2003 American Megatrends, Inc.
BIOS Version:HDAMC222   Date:04/13/2005
CPU : AMD Engineering Sample
 Speed : 2.2 GHz    Count : 4
DRAM Clocking CPU0 Core0/1 = 266 MHz, CPU1 Core0/
1024MB Physical RAM
Press DEL to run Setup
Press F11 for BBS POPUP
2 AMD North Bridges, Rev E1
Checking NVRAM..
1024MB OK


 (C) American Megatrends, Inc.
63-0100-000005-00101111-041305-OPTERON-HDAMC222-Y
```



```
GRUB version 0.95 (638 lower / 260032K upper memory)

 Debian 2.6 Kernel
 Debian 2.4 Kernel

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.
```



Copyright © 1985-2001
Microsoft Corporation



```
Calibrating delay loop... 2510.02 BogoMIPS
Memory: 60652k/65536k available (910k kernel code, 4496k reserved, 372k data, 72
k init, 0k highmem)
Dentry cache hash table entries: 8192 (order: 4, 65536 bytes)
Inode cache hash table entries: 4096 (order: 3, 32768 bytes)
Mount-cache hash table entries: 1024 (order: 1, 8192 bytes)
Buffer-cache hash table entries: 4096 (order: 2, 16384 bytes)
Page-cache hash table entries: 16384 (order: 4, 65536 bytes)
CPU: L1 I Cache: 64K (64 bytes/line), D cache 64K (64 bytes/line)
CPU: L2 Cache: 256K (64 bytes/line)
CPU: AMD Athlon(tm) Processor stepping 04
Enabling fast FPU save and restore... done.
Checking 'hlt' instruction... OK.
Checking for popad bug... OK.
POSIX conformance testing by UNIFIX
PCI: PCI BIOS revision 2.10 entry at 0xfd980, last bus=0
PCI: Using configuration type 1
PCI: Probing PCI hardware
PCI: Using IRQ router PIIX [8086/7110] at 00:07.0
PCI: Cannot allocate resource region 4 of device 00:07.1
Limiting direct PCI/PCI transfers.
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
_
```

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`)

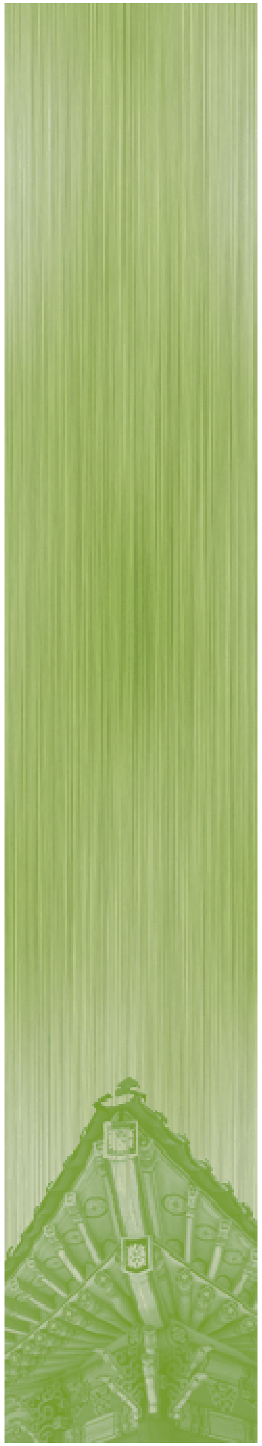
System Startup (4)

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 week, ~9/30)
- **Project 1**
  - Threads (2 weeks, ~10/15)
- **Project 2**
  - User programs (3 weeks, ~11/5)
- **Project 3**
  - Virtual memory (5 weeks, ~12/10)
- **This schedule is subject to change**

Project Policy (1)

- **Team project (except Project 0)**
  - Three members in a team
  - You must work in teams in the “real world”
  - Communicate with colleagues (team members)
    - Communication problems are natural
    - It’s a good chance to get to know each other
    - How to divide work among team members?
    - What have you done?
    - What answers you need from others?
    - You must document your work!
    - You should clearly state the contribution of each team member in your project report
      (And this should be agreed upon among team members)

Project Policy (2)

- **Working in teams**
  - Do not try to merge all the codes developed independently by each team member just before the deadline
  - Often two changes conflict with each other, requiring lots of debugging
  - Instead, integrate your team's changes early and often.
  - Understand your requirement first. And then design well before the actual implementation

    → This will save your time considerably.
  - Refer to 2.1.4: Development Suggestions

Project Policy (3)

- **Late policy**
  - Each team has 5 “slip” days
  - 20% off per day after slip days exhausted
  - No advantage on remaining slip days
  - Save your slip days for rainy days, as the project is getting harder and harder

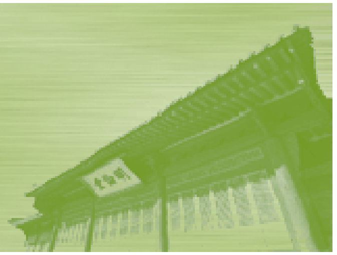
  - For Project 0, there is no slip day.

Project Policy (4)

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 0 points in the particular project and the final grade will be lowered by one grade (e.g., B+ → B)
  - For serious offenses, you will get an F grade and 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!

Project Grading (1)



- **Presentations in the Lab session (bonus)**
- **Functionality (70%)**

```
$ make check
$ make grade
```

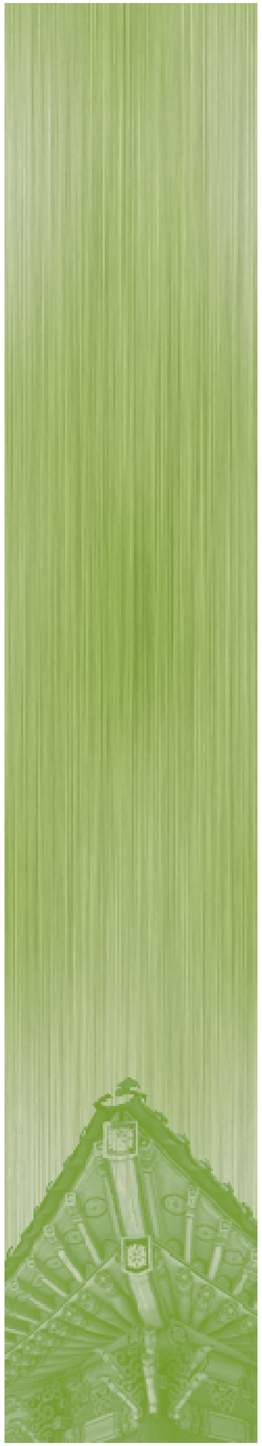
- **Design & documentation (30%)**
  - Source code
  - Design document
    - Data structure, Algorithm, Synchronization, Rationale
  - Refer to Appendix D: Project Documentation
- **Demos & oral tests**

Project Grading (2)

- **Demos & oral tests**
  - Usually done in the next week of the due date
  - Each team should meet the instructor offline
  - All team members should be present
  - You may bring your notebook as there could be a problem in running your solution in the instructor's machine
  - You should be able to answer any questions on
    - Basic system architecture
    - Design decisions
    - Implementation details
    - …

Project Grading (3)

- **Individual score**
  - = *f* (overall project score, individual contribution)
  - You should specify the followings in your report:
    - The percentage of contribution for each team member
    - The detailed list of specific tasks done by each team member
  - The report should be signed by all team members as a token of acceptance.
  - During demos & oral tests, the percentage of contribution can be adjusted by the instructor.
  - As long as your contribution is >= 25%, you will get the full project score.



Project 0: Warming Up

Project 0 (1)

- **Set up your own project environment**
  - Install Linux
  - Install all the required tools
  - Install Pintos
  - Capture the screen shot of working Pintos

```
$ pintos run alarm-multiple
```

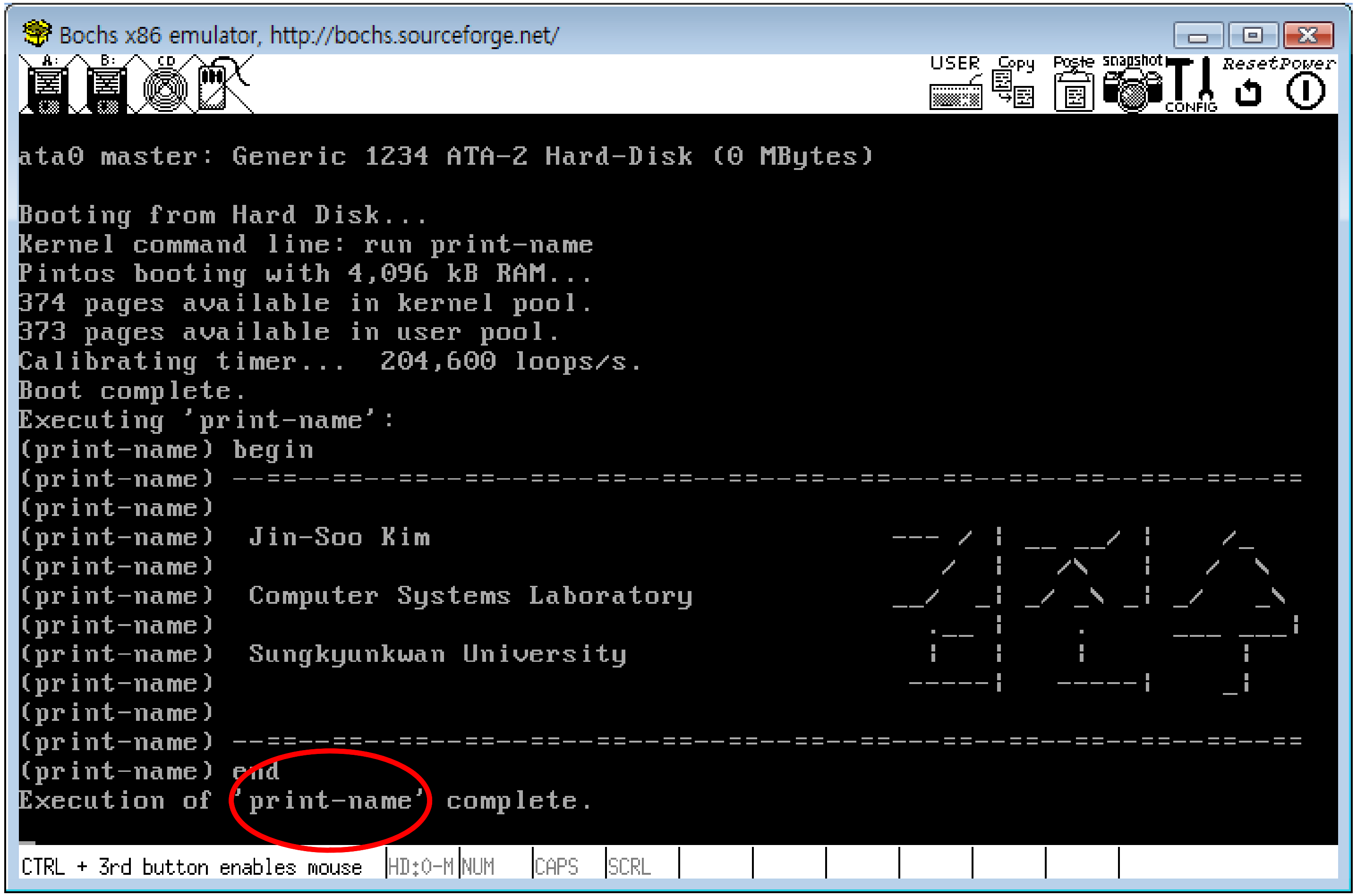
Project 0 (2)

- **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
  - Be creative when you print your name!
  - Capture the screen shot

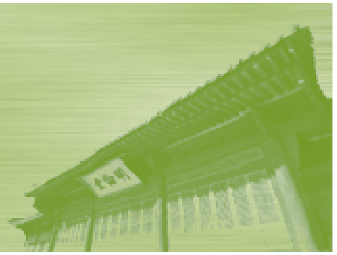
Project 0 (3)

- **Example:**


Bochs x86 emulator, http://bochs.sourceforge.net/
A:
B:
CD
USER
Copy
Paste
snapshot
CONFIG
Reset
Power
CTRL + 3rd button enables mouse
HD:0-M
NUM
CAPS
SCRL

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

Booting from Hard Disk...
Kernel command line: run print-name
Pintos booting with 4,096 kB RAM...
374 pages available in kernel pool.
373 pages available in user pool.
Calibrating timer...  204,600 loops/s.
Boot complete.
Executing 'print-name':
(print-name) begin
(print-name) --==--==--==--==--==--==--==--==--==--==--==--==--==--==--==--==
(print-name)
(print-name)  Jin-Soo Kim                            --- / | __ __/ |    /_
(print-name)                                            /  |   /\   |   /  \
(print-name)  Computer Systems Laboratory            __/  _| _/ _\ _| _/   _\
(print-name)                                           .__ |    .     ___ ___|
(print-name)  Sungkyunkwan University                  |   |    |         |
(print-name)                                          -----|   -----|    _|
(print-name)
(print-name) --==--==--==--==--==--==--==--==--==--==--==--==--==--==--==--==
(print-name) end
Execution of 'print-name' complete.
```

Project 0 (4)



Documentation

- Specification of your environment
  - Linux distributions, versions of gcc, etc.
- 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

Due:

- Sep. 30, 11:59PM (NO slip day)
- Submit via e-mail to jinsookim@skku.edu
- Note: This is an individual project