CS143A
Principles of Operating Systems
Discussion 01: Project Setup

Instructor: Prof. Ardalan Amiri Sani
TA: Ping-Xiang Chen (Shawn)

Acknowledgement

The slides are based on the previous discussions from Dr. Saehanseul Yi.

About me

- 4th year CS PhD student
- Research interests: Optimizing I/O stacks for emerging storage devices
- Email: p.x.chen@uci.edu
- Office hour: Thursdays from 9:30-10:30 am (ICS 458A)

Agenda

- Remote development environment
- Brief introduction to Linux system
- Project setup demo

Project

- Lab0 & Lab1
- Pintos: a simple operating system
- 32-bit x86 emulators: Bochs vs. qemu
- Required programs
  - Remote connection: terminal or putty, X11 client(for GUI)
  - Development tools: make, gcc, gdb, …
  - Source code editors: vim, Visual Studio Code, ...

Remote Development Environment

IP Address
169.29.35.66

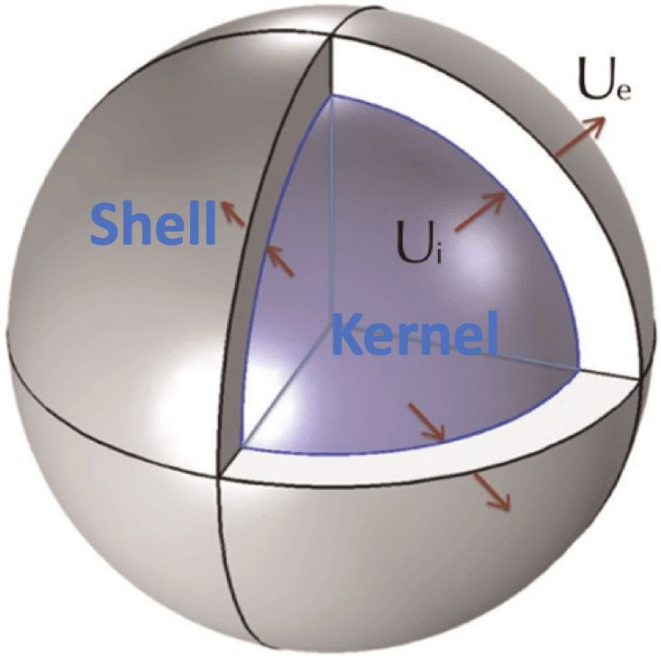
Over-simplified internet

Domain: openlab-1.xxx.edu

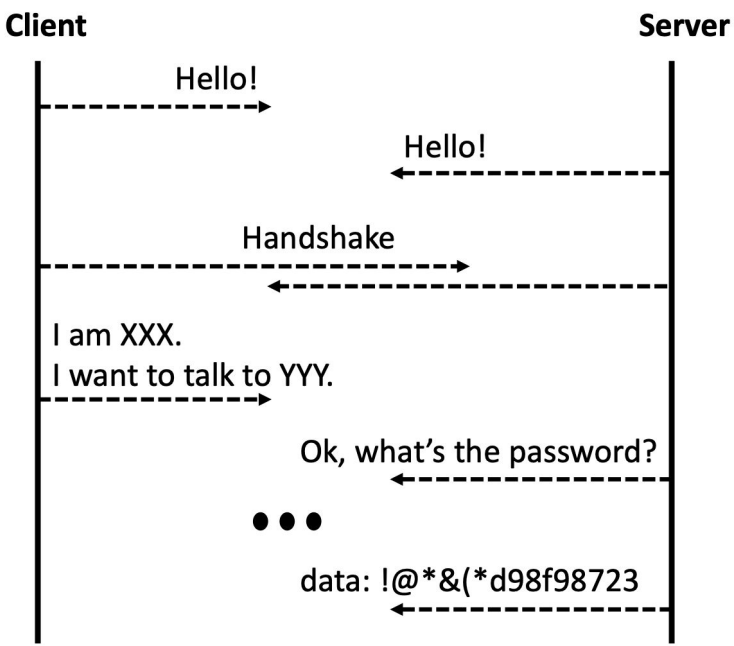
ICS Server (Nodes)

Remote Development Environment

- For the OS in each node,
  - The core of the OS is kernel
  - Kernel is responsible for fairly distributing resources to multiple users (or programs)
  - Users submit requests via shell (shell ≈ terminal ≈ console)
  - There is one kernel, but could be multiple shells (for each user)
  - Can we talk to a shell remotely?

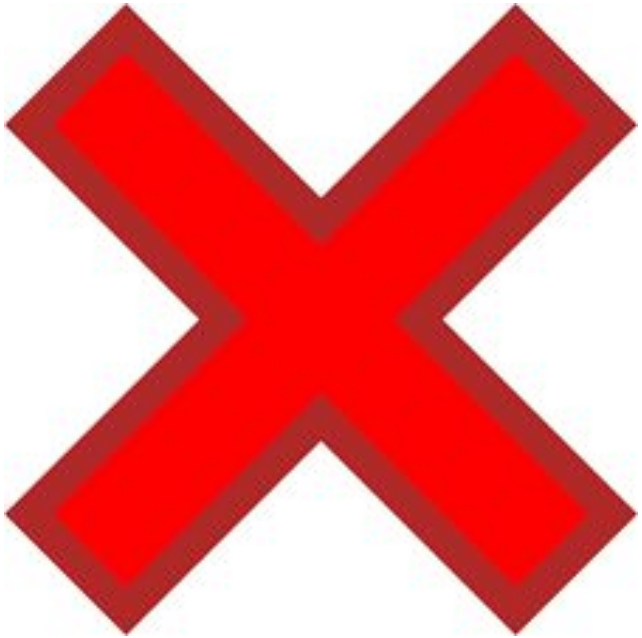


Remote Development Environment: Protocols



- Frequently used protocols:
  - **SSH** (Secure Shell Protocol) : characters
  - **X11**: graphical stuff
  - **FTP** (File Transfer Protocol): files
  - **SFTP** (Secure File Transfer Protocol): files
  - **SCP** (Secure Copy Protocol): files

Work Locally (Your computer)



Work Remotely (using UCI Openlab)

Windows	MacOS	Linux
Putty or WSL + Xming (for putty, make sure X11 Forwarding is checked in option)	built-in terminal + Xquartz (ssh with –X option)	built-in terminal (ssh with –X option)

Microsoft Visual Studio Code (VS Code)

Work Remotely (using UCI Openlab)

- $ ssh UCInetID@openlab.ics.uci.edu -X
- Passwords are invisible. Just type it
- Case matters, “A” and “a” are different
- If you have login problems please visit:
  - https://www.ics.uci.edu/~lab/students/

Welcome to Linux!

- /: root directory
- The “path” always starts with /
- In a path, directories are separated with /
- After login, you will be at your home directory: /home/UCINetID
- First command:
  - pwd (Print Working Directory)

```
pingxiac@circinus-4 21:43:03 ~
$ pwd
/home/pingxiac
```

Welcome to Linux!

- Shell types: GUI vs. CUI
  - Character/graphical user interface
  - CUI has its own advantages over GUI and used very widely these days
- Basics of CUI
  - Users are given a prompt to type a command (usually a $ sign)
  - Then you enter a command and its arguments. ($ cp a.txt b.txtècopy a.txt into b.txt)
  - Each of these “commands” is actually a program stored in a pre-defined directory
  - E.g., to open chrome, double click the icon OR type “chrome” in a CUI shell

Welcome to Linux!

- Pre-defined directory? Where is it stored?
- Environment variables (env vars)
  - volatile variables that are used by shell
  - PATH=/bin:/usr/bin:/usr/sbin programs here can be executed by its name
  - SHELL=/bin/bash
  - PWD=/home/pingxiac
- Volatile?
  - Any modification to these variables that you want to save should be stored in a file (~/.bashrc)
  - Otherwise, it will be reset to default.

Welcome to Linux!

- man <command>: manual for the command
- E.g. man pwd

```
PWD(1)                                                    User Commands

NAME
       pwd - print name of current/working directory

SYNOPSIS
       pwd [OPTION]...

DESCRIPTION
       Print the full filename of the current working directory.

       -L, --logical
              use PWD from environment, even if it contains symlinks

       -P, --physical
              avoid all symlinks

       --help display this help and exit

       --version
              output version information and exit

       If no option is specified, -P is assumed.
```

Pintos Project Setup (1/7)

- Create a directory
  - $ mkdir Pintos

Linux command: file handling

Command	Short for…	Description
mkdir <dir_name>	make directory	
touch <file_name>		create an empty file
mv <source> <dest>	move	move files(dirs.) or rename
cp <source> <dest>	copy	copy files(dirs.) + rename
rm <file_name>	remove	remove file
rm –r <dir_name>	remove recursively	remove directories

Note: rm is not reversible; no way to recover the files! Be careful

Pintos Project Setup (2/7)

- Get Pintos source code
  - $ cd Pintos
  - $ git clone https://github.com/trusslab/pintos.git

- ./ (dot followed by a slash): means the current directory (relative path).
- An absolute path is the path starts from the root directory. i.e. /home/UCNetID

Linux command: Navigation

Command	Short for…	Description
pwd	Print Working Directory	Current working directory
ls	List	List files and directories
cd	Change directory	go to home directory
cd ..		go out to parent directory
cd <directory_name>		go inside the directory

Pintos Project Setup (3/7)

- Make an empty directory for Bochs
  - $ mkdir bochs
  - we are at ~/Pintos

Pintos Project Setup (4/7)

- Build Bochs
  - $ cd pintos/src/misc/
  - $ ./bochs-2.6.2-build.sh ~/Pintos/bochs
- File extensions are not strictly required in Linux systems
- Though, we often put extensions to easily identify files
- .sh here implies ‘shell script’; it executes a series of commands for building Bochs: downloading source code, build, patch bugs, ...

Pintos Project Setup (5/7)

- Build Pintos utilities
  - $ cd ~/Pintos/pintos/src/utils/
  - $ make
- make is a program for building executables from source code
- it uses a file called makefile which contains a set of rules for building

Pintos Project Setup (6/7)

- Directories for executables
  - $ cd ~/Pintos/pintos
  - $ mkdir bin
  - $ mkdir misc
  - $ cd ~/Pintos/pintos/src/utils
  - $ cp backtrace pintos* Pintos.pm setitimer-helper squish-* ~/Pintos/pintos/bin/
  - $ cp ~/Pintos/pintos/src/misc/gdb-macros ~/Pintos/pintos/misc/

Pintos Project Setup (7/7)

- Update environment variables
  - $ vi ~/.bashrc
- Add the following to ~/.bashrc
  - export PATH=$PATH:~/Pintos/pintos/bin
  - export PATH=$PATH:~/Pintos/bochs/bin
- Then
  - $ source ~/.bashrc

Verifying Pintos Project Setup

- $ which pintos
- $ which bochs
- Unsuccessful
  - $ which pintos
    - /usr/bin/which: no pintos in (/home/pingxiac/....)
- Successful
  - $ which pintos
    - ~/Pintos/pintos/bin/pintos

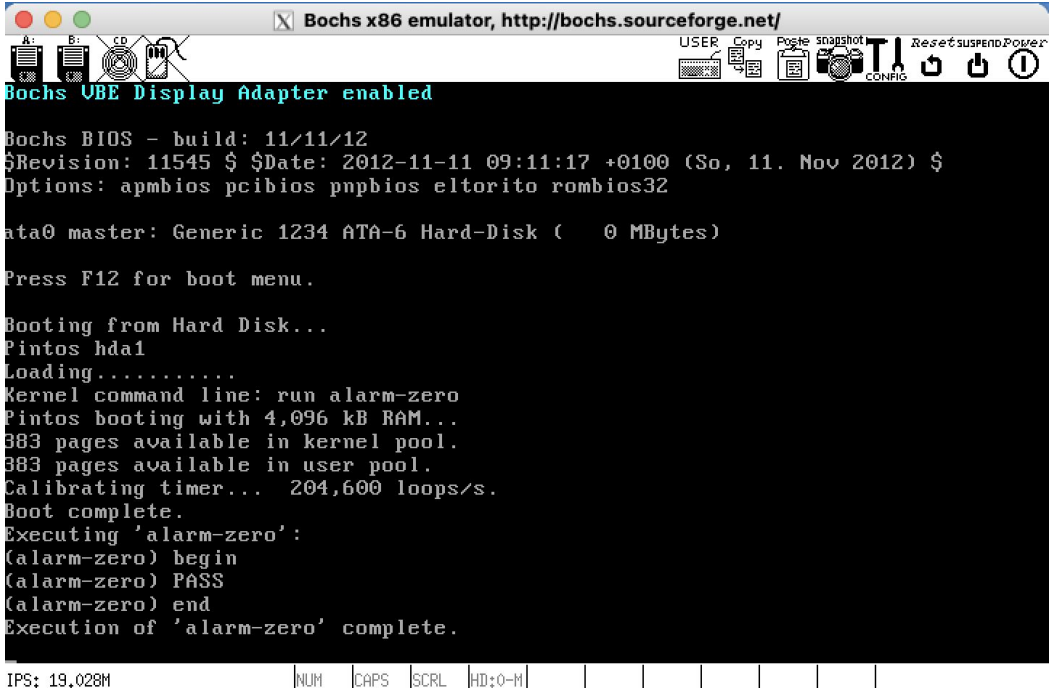
Verifying Pintos Project Setup

- Directory/file structure check

```
pingxiac@circinus-2 22:37:42 ~
$ tree -L 3 Pintos/
Pintos/
├── bochs
│   ├── bin
│   │   ├── bochs
│   │   ├── bochs-dbg
│   │   ├── bxcommit
│   │   └── bximage
│   └── share
│       ├── bochs
│       ├── doc
│       └── man
└── pintos
    ├── bin
    │   ├── backtrace
    │   ├── pintos
    │   ├── pintos-gdb
    │   ├── pintos-mkdisk
    │   ├── Pintos.pm
    │   ├── pintos-set-cmdline
    │   ├── setitimer-helper
    │   ├── squish-pty
    │   └── squish-unix
    ├── misc
    │   └── gdb-macros
    ├── README.md
    └── src
        ├── devices
        ├── examples
        ├── filesys
        ├── lib
        ├── LICENSE
        ├── Make.config
        ├── Makefile
        ├── Makefile.build
        ├── Makefile.kernel
        ├── Makefile.userprog
        ├── misc
        ├── tests
        ├── threads
        ├── userprog
        ├── utils
        └── vm
```

Booting Pintos

- $ cd ~/Pintos/pintos/src/threads
- $ make
- $ cd build
- $ pintos --bochs -- run alarm-zero
  - (or pintos -v --bochs -- run alarm-zero)
  - With -v option, it will be verbose, no additional windows
  - To quit, hit Ctrl + c (default shortcut for canceling tasks in Linux)



Pintos, Infinite Loop?

```
$ pintos --bochs -- run alarm-zero
squish-pty bochs -q
========================================================================
                       Bochs x86 Emulator 2.6.2
               Built from SVN snapshot on May 26, 2013
                 Compiled on Jan 10 2023 at 16:28:31
========================================================================
00000000000i[     ] reading configuration from bochsrc.txt
00000000000e[     ] bochsrc.txt:8: 'user_shortcut' will be replaced by new 'keyb
oard' option.
00000000000i[     ] installing nogui module as the Bochs GUI
00000000000i[     ] using log file bochsout.txt
Pintos hda1
Loading...........
Kernel command line: run alarm-zero
Pintos booting with Pintos hda1
Loading...........
Kernel command line: run alarm-zero
Pintos booting with Pintos hda1
Loading...........
Kernel command line: run alarm-zero
Pintos booting with Pintos hda1
```

- Pintos is an old program, so not compatible with latest toolchains installed on Openlab
- The toolchain build instructions on the course webpage needs an update
- For convenience, we are distributing pre-built toolchains

Pintos, Infinite Loop?

- In your home folder (/home/YOUR_UCINET_ID)
- $ wget http://www.ics.uci.edu/~ardalan/courses/os/pintos-toolchains.tgz
- $ tar -xvf pintos-toolchains.tgz
- (add this line in your ~/.bashrc)
  - export PATH=/home/YOUR_UCINET_ID/pintos-toolchains/x86_64/bin:$PATH
- (the last ":$PATH" is extremely important)
- Exit and reconnect
- If you have previously built Pintos, go to threads directory (~/Pintos/pintos/src/threads) and remove build directory (rm -rf build)
- Type make again

How to debug? Read here!

- E.1 printf()
- E.2 ASSERT
- E.3 Function and Parameter Attributes
- E.4 Backtraces
- E.5 GDB

The fatal python error

- $ pintos-gdb
  - Fatal Python error: _PyOS_InterruptOccurred: the function must be called with the GIL held, but the GIL is released (the current Python thread state is NULL) Python runtime state: unknown
- change the content of: /home/UCInetID/Pintos/pintos/bin/pintos-gdb

```
#! /bin/sh

# Path to GDB macros file.  Customize for your site.
GDBMACROS=$(dirname $0)/../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
```

gdb

How to use GDB?

- GDB, or the GNU Debugger, is a powerful debugger that allows you to step-by-step execute a program.
- start Pintos with the --gdb option (terminal 1)
  - $ pintos --bochs --gdb -- run alarm-zero
- Open another terminal
  - Make sure both GDB and pintos are running on the same machine by running hostname in each terminal.
- Go to build directory to find the built kernel.o (terminal 2)
  - $ cd ~/Pintos/pintos/src/threads/build
- Use pintos-gdb to invoke GDB on kernel.o (terminal 2)
  - $ pintos-gdb kernel.o --tui
  - $ debugpintos
- TUI option means invoke GDB Text User Interface
  - More information about GDB Text User Interface
- Now, you are able to use GDB to debug Pintos

Lab0: Kernel Monitor

- Standard C library functions (printf, scanf, ...) are often unavailable in kernel-level programming (printf is provided by pintos)
- In Pintos, there often exists a low-level alternative for those functions
- For scanf, check out input_getc in devices/input.c
- Please be aware
  - The result of whoami command should only contains upper- and lower-case letters.

Project Submission (1/4)

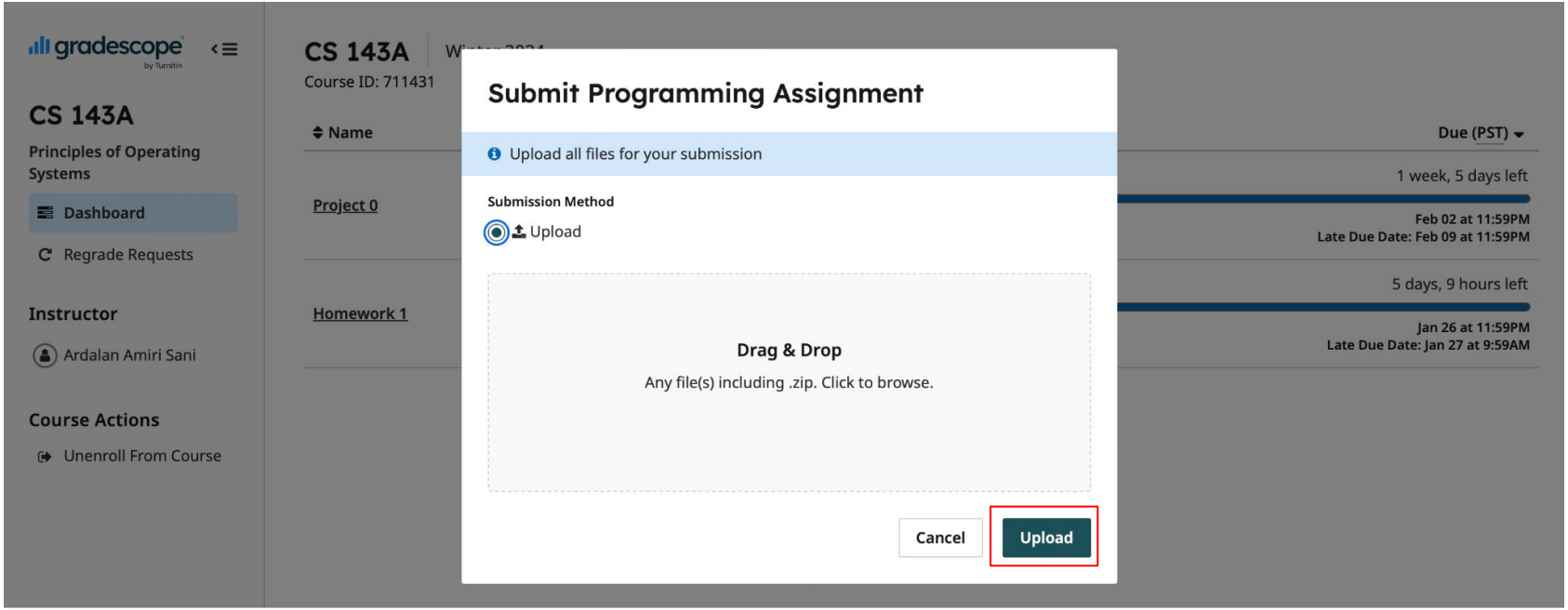
- The source code should also contain your screenshot and design doc in the folder
  - ~/Pintos/pintos/src/p0
- Compress the pintos source code with your modification
  - $ cd ~/Pintos
  - $ tar -zcvf pintos.tar.gz pintos

Project Submission (2/4)

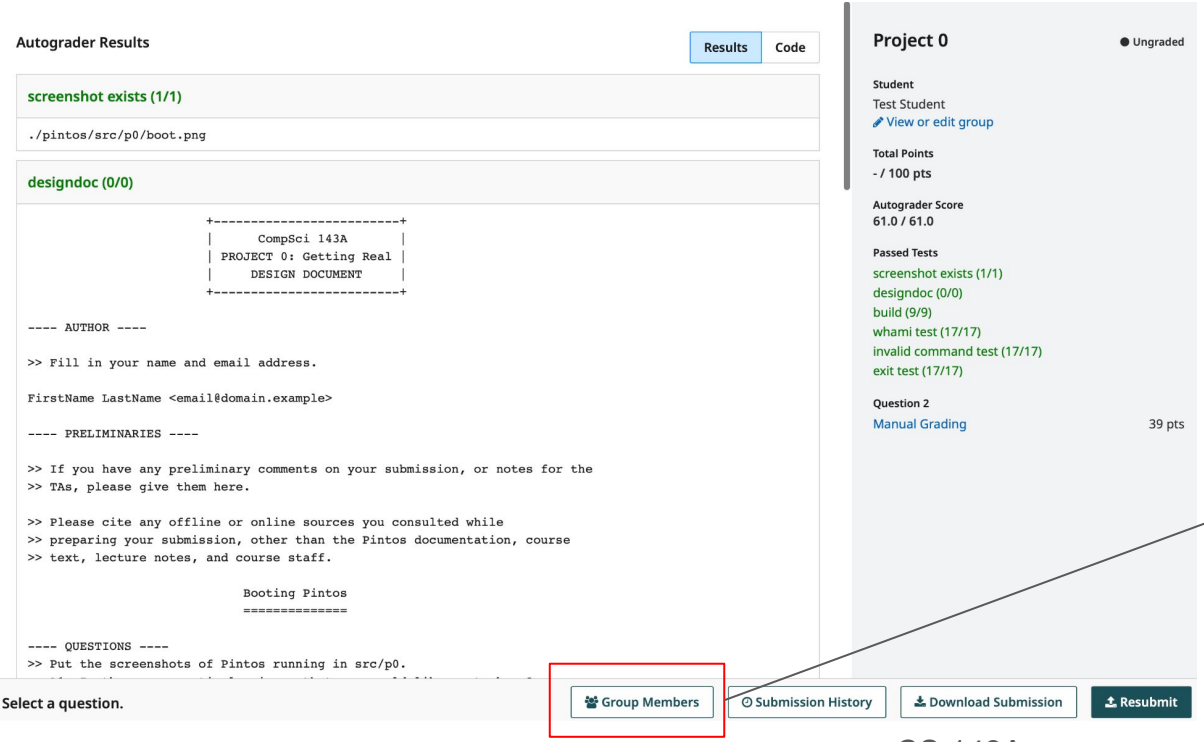
- Copy your compressed project to your laptop with SCP (Secure Copy Protocol:
  - $ scp UCInetID@openlab.ics.uci.edu:/home/UCInetID/Pintos/pintos.tar.gz taget_folder_in_your_local_comouter

Project Submission (3/4)

- Upload your project 0 to gradescope



Project Submission (4/4)



- The grading result will show up after a short period of time.
- Project grading contains 2 parts
  - Autograder: your code correctness
  - Manual grade: TAs will grade your design doc manually.

You can edit your group members of your submission. (maximum 3 students per group)