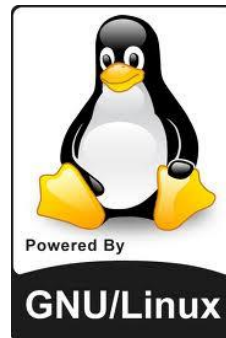


# Welcome to Linux world

Signed-off-by: GUAN Xuetao <gxt@mprc.pku.edu.cn>



# Course Perspective

## ■ Most Systems Courses are Builder-Centric

- Computer Architecture
  - Design pipelined processor in Verilog
- Operating Systems
  - Implement sample portions of operating system
- Compilers
  - Write compiler for simple language
- Networking
  - Implement and simulate network protocols

# Course Perspective (Cont.)

## ■ Our Course is Programmer-Centric

- Purpose is to show that by knowing more about the underlying system, one can be more effective as a programmer
- Enable you to
  - Write programs that are more reliable and efficient
  - Incorporate features that require hooks into OS
    - E.g., concurrency, signal handlers
- Cover material in this course that you won't see elsewhere
- Not just a course for dedicated hackers
  - **We bring out the hidden hacker in everyone!**

# Policies: Labs

## ■ Work groups

- You must work alone on all lab assignments

## ■ Handins

- Labs due at 11:59pm on Monday or Wednesday
- Electronic handins using **Autolab** (no exceptions!)

## ■ Labs will use the Linux Servers

- `linux> ssh -p xx22 student-id@ics9.pku.edu.cn`
- 8 servers
  - 1 gateway machine, 3~4 student machines (for student logins)
  - 1 head node (for Autolab server)
  - 1 grading machines (for autograding)
- Each server:
  - 8 Intel Xeon E5-2650 (32 cores)
  - 32 GB DRAM
  - Ubuntu 16.04 amd64

# Timeliness and version limit

## ■ Grace days

- **5 grace days** for the semester
- **Limit of 2 grace days** per lab used **automatically**
- Covers scheduling crunch, out-of-town trips, illnesses, minor setbacks
- Save them until late in the term!

## ■ Lateness penalties

- Once grace day(s) used up, get penalized **10% per day**
- No handins later than **3 days after due date**

## ■ Late Slack: 3600

- This is the number of seconds after a deadline that the server will still accept a submission and not count it as late.

## ■ Default Version Threshold: 16

- If a submission's version is greater than this threshold, it is penalized according to the version penalty.

## ■ Default Version Penalty: **~10% points**

- The penalty applied to submissions with versions greater than the version threshold.

# 内容

- Linux入门
- Linux编程与工具
- Autolab简介

# Linux入门

师傅领进门，修行在个人

# Shell

- **Shell definition: interface between the user and the operating system**
  - fully programmable, interpreted
  - will read from standard input or file input; interacting with system vs. shell programming
- **Different shells (targeted toward different uses and applications)**

■ Bourne (sh)	original UNIX shell
■ Korn (ksh)	superset of Bourne
■ C (csh)	has a C-like syntax
■ Bash (bash)	bourne again shell, superset of Bourne
■ tcsh, zsh, ...	
■ develop your own!	
- **Environmental variables: HOME, USER, PS1**



# Accessing your Linux account

## ■ SSH: Secure SHell

## ■ Login process

- login name echo'ed
- password not echo'ed
- if you enter an invalid string for either, the system will not indicate which was invalid

## ■ Some system status commands:

- date, hostname, whoami (or logname), who, w, uptime (when was the system last rebooted), uname
- the password are changed by execution of the passwd utility program

## ■ Logging out

- hit <ctrl-d> or enter exit

**Change your password: passwd**

# SSH: Secure SHell

- a common protocol for connecting to remote computers
- several free versions of SSH available
  - My personal favorite is PuTTY
- **File transfer**
  - scp
  - sftp

Windows和Linux互传文件的问题:  
权限和文本换行符

# Kernel *metacharacters*

- **Priority: shell does not receive command line until kernel has interpreted all kernel *metacharacters***
  - erase (^?): deletes previous character
  - kill (^U): deletes entire line
  - eof (^D): end of file
  - intr (^C): kills current foreground process
  - suspend (^Z)
  - stop (^S)
  - start (^Q)
  - and others
- **stty -a (displays them)**
  - get/set kernel *metacharacters*
- **tset/reset**
  - terminal initialization

# General syntax of UNIX commands

■ `<command> [<options>] [<argument(s)>]`

■ examples:

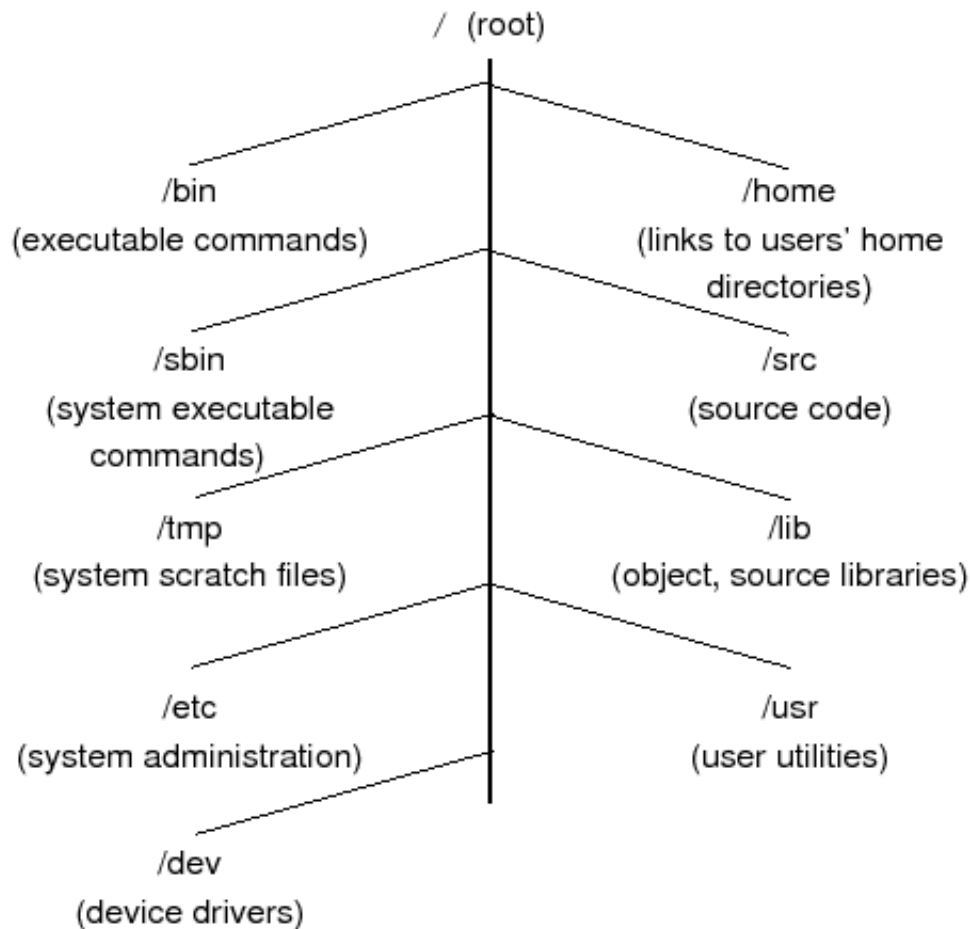
- `$ ls`
- `$ ls -l`
- `$ ls -l myfile`
- `$ ls -ld mydir`
- `$ ls -F mydir`
- `$ ls -al`
- `$ ls -a -l (POSIX)`
- `$ ls -l -d mydir`

■ command names (like filenames) are case sensitive

# Getting help on the UNIX system

- **For a help on a particular command, use `man <command>`**
  - `man` retrieves the manpage (manual page) for any command (and functions in C libraries)
  - e.g., `man wc`, `man -s 3C printf`, or `man fgetc`
  - `man man` (eerie self-referential command) (*eerie: weird, mysterious, supernatural*)
- **For all commands on a general topic, use `apropos`**
  - `apropos <keyword/topic>`, e.g., `apropos copy`
  - `apropos = man -k`
- **Examples:**
  - `whatis = man -f <title>`
  - manpage can be searched with `/<keyword/topic>`
  - `man printf` (which section?)
  - use `man -a printf` (all)
  - `man -s 2 fork`
  - `man -s 3 intro`

# UNIX Filesystem



- Abstract: file
- highest level directory = root /
- working directory

# Absolute vs. relative path

- **pathname: filename preceded by directories leading to the file**
  - absolute pathname: the complete pathname of a file starting with the root /
  - relative pathname: pathname which implicitly starts at the working directory
- **two special files in every directory**
  - . is a link to the current working directory
  - .. is a link to the directory containing the current working directory (i.e., its parent)

# File manipulation and management

## ■ Navigating through directories: cd

## ■ File manipulation and management

- pwd: print working directory
- cp <src> <dest>: copy file(s)
- rm <file(s)> (remove): erases file(s)
- mkdir <dir(s)> (make directory): creates one or more directories
- rmdir <dir(s)> (remove directory): deletes one or more directories
- mv <src> <dest> (move): relocates the file(s) into the specified directory
- more/less <file(s)> : display <file(s)>
- head, tail: can specify number of lines with option
- chmod <mode> <file(s)> : change file mode bits



# Linux编程与工具

工欲善其事，必先利其器

# Hello world!

- Edit hw.c

- Compile hw.c

- Run hw

```
#include <stdio.h>

int main()
{
    printf("Hello world!\n");
}
```

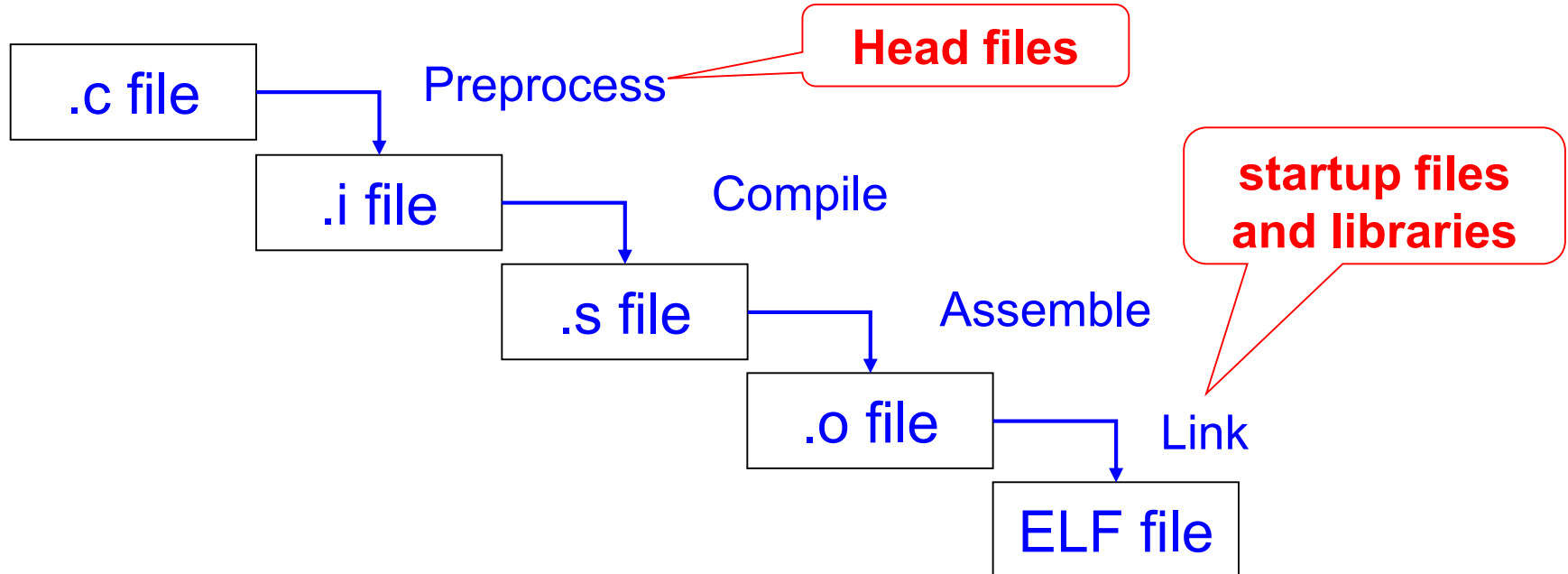
```
$ gcc -o hw hw.c
$
```

```
$ ./hw
Hello world!
$
```

# Examine compilation details

```
$ gcc -o hw -save-temps hw.c
$ ll
total 52
drwxr-xr-x 2 guanxuetao guanxuetao 4096 Jul 25 13:52 ./
drwxr-xr-x 3 guanxuetao guanxuetao 4096 Jul 25 10:44 ../
-rwxr-xr-x 1 guanxuetao guanxuetao 7546 Jul 25 13:52 hw*
-rw-r--r-- 1 guanxuetao guanxuetao 63 Jul 25 10:45 hw.c
-rw-r--r-- 1 guanxuetao guanxuetao 21670 Jul 25 13:52 hw.i
-rw-r--r-- 1 guanxuetao guanxuetao 904 Jul 25 13:52 hw.o
-rw-r--r-- 1 guanxuetao guanxuetao 415 Jul 25 13:52 hw.s
$
```

# Building flow



# Examine the code

`ldd hw`

examine library dependency

`nm hw`

examine symbol name definitions

**so many unfamiliar symbols**

find “`T main`” near at the bottom

# Brief anatomy of ELF

## Executable and Linking Format

```
file hw
```

```
readelf -a hw
```

```
readelf -S hw
```

```
strip hw
```

```
ls -l
```

```
readelf -S hw
```

```
nm hw
```

```
ldd hw
```

```
./hw ; echo $?
```

**Discard symbols  
from object files**

# the vi editor

- **moded editor**

- **two modes**

- type i to enter insert mode
- hit <escape> to enter command mode
  - by default in command mode
  - use {h,j,k, l} keys to move {left, down, up, right}, resp., why?
  - u key undoes the previous operation
  - :w (file write)
  - :q (quit, no write)
  - :wq = <shift-ZZ> (file write and quit)

- **\$ vimtutor**

- **quick reference card/guide**

# GNU Tools

- **gcc**
  - GNU Project C and C++ Compiler
- **make**
  - GNU make utility to maintain group of programs
- **gdb**
  - The GNU debugger



# GCC

## ■ gcc options

- -c Create symbolic link for each src file
- -o filename Specify output file as file
- -x language Expect input file to be written in language
- -v Verbose mode
- -S Compile source file into assembler code
- -E Preprocess source file, do not compile
- -O[level] Optimize level, 1, 2, 3, 0
- -g Include debug information as user use gdb

## ■ Binutils

- objdump, strings, readelf, nm, ...

# Debugging with gdb

## ■ GDB -- the GNU debugger

- Invoked with “gdb [program]” or “gdb --args [program] [arg1] [arg2] ...”
- To allow gdb access to information about your C source, compile with “gcc -g”
- Allows you to step through your code, examine program state

## ■ Breakpoints – tell gdb to “Run until you get to this point”

- In gdb, type “break [argument]”
- Function names
- Line numbers (with -g)
- Addresses – e.g. “break \*0x00000e80”

## ■ Running your program

- start
  - Loads your program, but pauses before running any code
- run
  - Loads your program and starts it executing
  - Runs until it terminates or hits a breakpoint

# Debugging – Moving through code

## ■ step/stepi

- step: Proceed to the next line of code, passing into any function call
- stepi: Proceed to the next instruction, passing into any function call
- Without -g, step doesn't know where the next line is
- Proceeds till program exits

## ■ next

- Only works with -g
- Proceeds to the next line of code, running through any functions required to get there

## ■ continue

- Runs the program until it hits a breakpoint or terminates

# Debugging – Examining your Program

## ■ print

- Takes an expression as its argument
- With -g, can use variable names
- Access registers with e.g. \$eax
- Can control formatting:
  - print/x for hex
  - print/t for binary
  - print/s for a null-terminated string
  - others – see “help print” for more information

## ■ examine

- Shows a memory location
- Approximately, “examine x” = “print \*x”

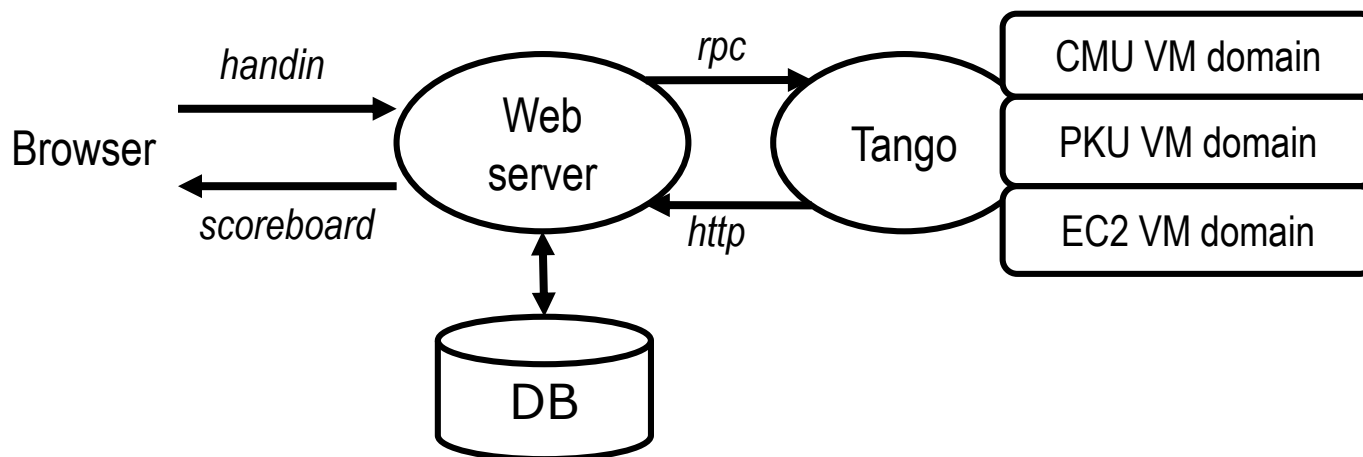
# Autolab简介

# Autolab Introduction

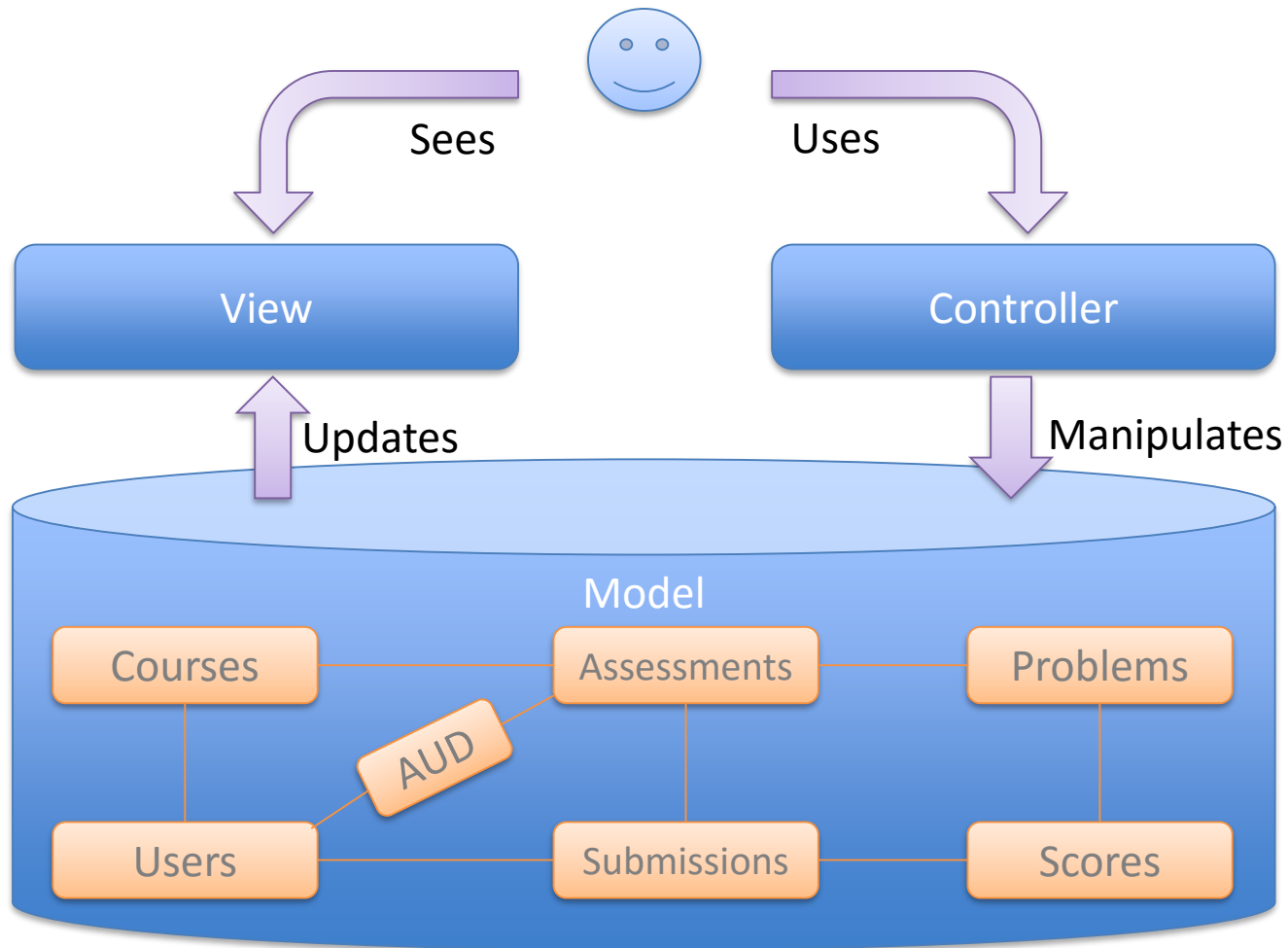
- **An online autograding service that allows instructors to offer programming assignments over the Internet.**
- **Two key ideas: autograding and scoreboards**
  - Autograding:
    - Programs evaluating the quality of other programs.
    - Student handins automatically autograded on secure VMs.
  - Scoreboard
    - Scores are posted in real-time on sorted class scoreboard.
    - Students anonymize themselves with nicknames.
      - “kill -9 ICS”, “ICS makes me ANSI”!
- **With Autolab you can use your Web browser to:**
  - Download the lab materials
  - Handin your code for autograding by the Autolab server
  - View the class scoreboard
  - View the complete history of your code handins, autograded results, instructor’s evaluations, and gradebook.
  - View the TA annotations of your code for Style points.

# Autolab系统组成

层次结构		基本功能
前端 Front-End	WebServer	用户交互、数据处理
	Database	数据管理、系统管理
后端 Back-End	Tango	任务管理、远程过程调用
	Tashi	集群管理器、节点管理器

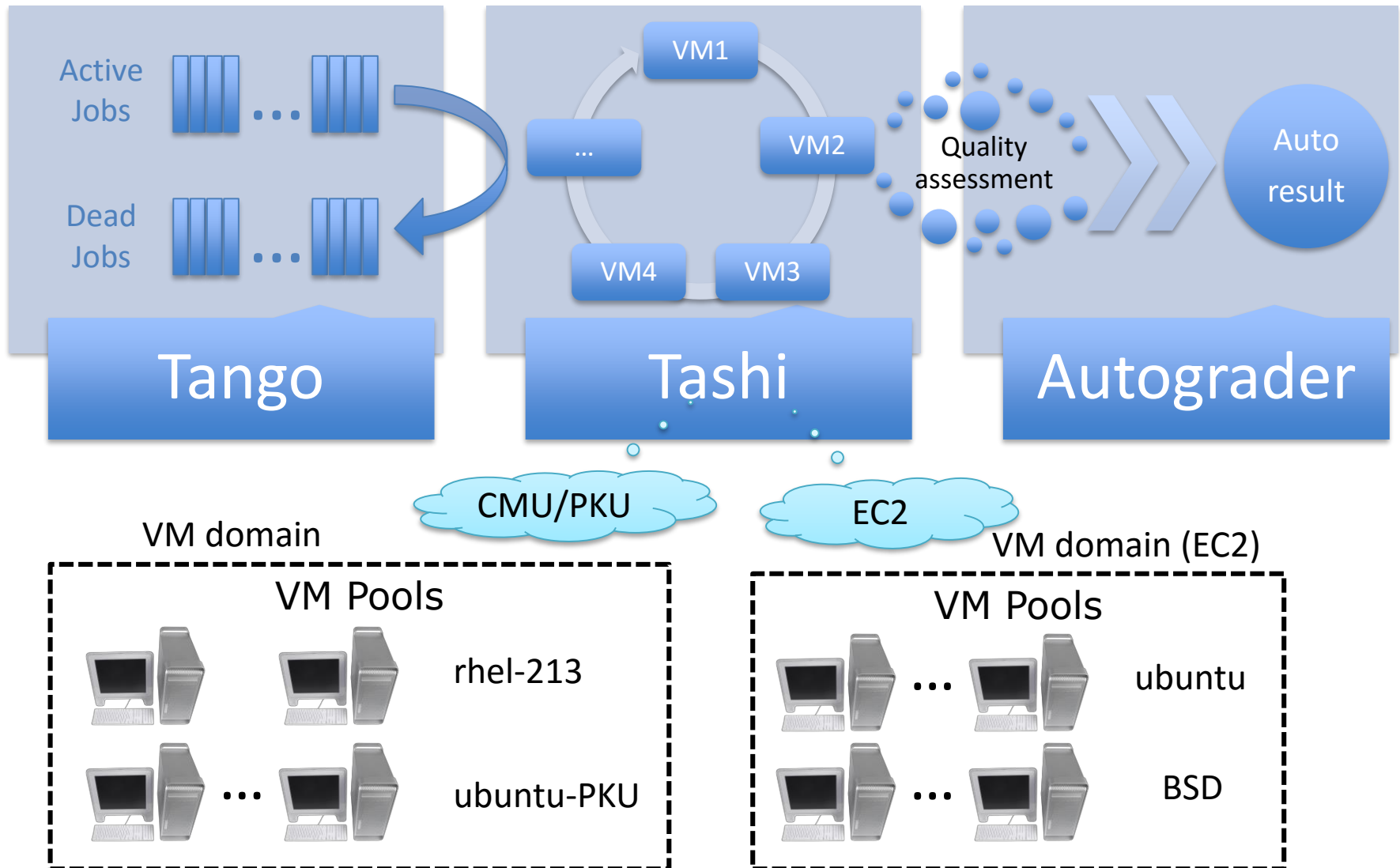


# 前端 ( Front-End )





# 后端 ( Back-End )



# Autolab: <https://ics16.pku.edu.cn/>

- **Account -> Change My Password**
- **Datalab -> View Writeup**
  - `datalab.pdf`
- **Datalab -> Download handout**
  - `datalab-handout.tar`
- **Datalab -> Handin your work**
  - `bits.c`
- **Due date and deadline**

# 八个实验

	名称	简要描述
L1	datalab	位级数据操作实验
L2	bomblab	拆解二进制炸弹实验
L3	attacklab	缓冲区溢出攻击实验
L4	archlab	处理器结构实验
L5	cachelab	性能优化实验
L6	tshlab	定制shell 程序实验
L7	malloclab	动态内存管理实验
L8	proxylab	Web 代理实验

# *Welcome and Enjoy!*

*人难免犯错误, 但要真把事情搞砸, 还需要一台电脑。*

《墨菲定律》

## *Just do IT!*