

VE280

Programming and Elementary Data Structures

Introduction; Linux

Logistics

- **Time:** Tuesday 10:00-11:40 am, Thursday 10:00-11:40 am, and Friday 10:00-11:40 am (on odd week).
- **Location:** East Middle Hall 1-400
- **Textbook Recommended (Not Required):**
 - “C++ Primer, 4th Edition,” by Stanley Lippman, Josee Lajoie, and Barbara Moo, Addison Wesley Publishing, 2005.
 - “Problem Solving with C++, 8th Edition,” by Walter Savitch, Addison Wesley Publishing, 2011.
 - “**Data Structures and Algorithm Analysis**,” by Clifford Shaffer. Online available:
<http://people.cs.vt.edu/~shaffer/Book/C++3e20120605.pdf>

Instructor

- Weikang Qian
- Email: qianwk@sjtu.edu.cn
- Phone: 3420-4020
- Office: Room 421, JI Building
- Office hour
 - Tuesday 12:50 – 1:50 pm
 - Thursday 12:50 – 1:50 pm
 - Or *by appointment*

Teaching Assistants

- Xu, Kejia
 - Email: kejiaxu@umich.edu
 - Cell phone: 15900908722



- Hao, Yifan
 - Email: pia1995@sjtu.edu.cn
 - Cell phone: 15026606710



Teaching Assistants

- Tao, Xuyang
 - Email: tao_xuyang@163.com
 - Cell phone: 13122212391



- Zhou, Hongkuan
 - Email: tedzhouhk@163.com
 - Cell phone: 15021382379



Teaching Assistants

- Yao, Yue
 - Email: patrickyao@sjtu.edu.cn
 - Cell phone: 15216699420



Grading

- Composition
 - In class quiz: 5%
 - (About) 5 programming projects: 50%
 - Midterm exam (written): 20%
 - Final exam (written): 25%
- We will assign grades on a curve, in keeping with past grades given in this course.
- Questions about the grading?
 - Must be mentioned to TAs or instructor within one week after receiving the item.

Projects

- Projects require:
 - Read and understand a problem specification
 - Design a solution (in your mind)
 - Implement this solution (simply and elegantly)
 - Convince yourself that your solution is correct

Projects

- We will give you a few simple test cases to get started. You should design your own set of tests (very important!).
- You will have chance to pre-test your program before the deadline.
 - We will use an online judge.
- Grading projects will be done by a combination of testing (correctness) and reading (implementation requirement and simplicity/elegance).

Programming Environment

- We require you to develop your programs on **Linux operating systems** using compiler `g++`.
- C++11 standard is allowed.
 - Compile with the option `-std=c++11`
- We will grade your programs in the Linux environment.
 - They must compile and run correctly on this operating system.

Project Deadline

- Each project will be given a due date. Your work must be turned in by 11:59 pm on the due date to be accepted for full credit.
- However, we still allow you to submit your homework within 3 days after the due date, but there is a late penalty.

Hours Late	Scaling Factor
(0, 24]	80 %
(24, 48]	60 %
(48, 72]	40 %

- No work will be accepted if it is more than 3 days late!

Project Deadline

- In **very occasional** cases, we accept deadline extension request.
 - Deadline extension requests will only be considered if you contact the course instructor in person. Do not contact TAs!
 - **ONLY** be granted for **documented** medical or personal emergencies that could not have been anticipated.
 - **NOT** granted for reasons such as accidental erasure/loss of files and outside conflicting commitments.

Some Suggestions

- Practice! Build demos yourself
 - You have the freedom. Even try something wrong on purpose
- Learn from your mistakes!
 - Take notes on the mistakes you make. Review frequently
- Start your project early!
 - Don't wait until the last minute. Numerous lessons before
- Make copies frequently in case your computer crashes.
 - Consequence: “computer crash” is NOT a reason for late submission!

Collaboration and Cheating

- You may discuss in oral with your classmates.
- **But** you must do all the assignments yourself.
- Some behaviors that are considered as cheating:
 - Reading another student's answer/code, including keeping a copy of another student's answer/code.
 - Copying another student's answer/code, in whole or in part.
 - Having someone else write part of your assignment.
 - Using test cases of another student.

“**Another student**” includes a student in the current semester or in the previous semester.

Collaboration and Cheating

- The previous lists of behaviors are deliberate cheating, but some unintentional actions could make you look like cheating. For example,
 - Testing your code with another one's account. Another's code may be overwritten by you. So, we see two identical copies.
 - You use another's computer to upload your code (in some cases like network/computer problems), but upload another's copy.
- We suggest you not to do those “dangerous” things.
 - If due to network/computer problem, you cannot upload, then send your code to TA's by email. By this way, you can double check the attachment.

Collaboration and Cheating

- In summary, you should be responsible for all answers/codes you submit. If you submit a copy of another student's work (or overwrite another student's work), it is considered cheating, **no matter of the reason!**

Collaboration and Cheating

- Any suspect of cheating will be reported to **the Honor Council at JI**.
- For programming assignments, we will run an automated test to check for unusually similar programs. Those that are highly similar - in whole or in part - will be reported to **the Honor Council at JI**.
- Penalty of honor code violation
 1. Reduction of the grade for this assignment to 0, **plus**
 2. Reduction of the final grade for the course by one grade point, e.g., B+ \rightarrow C+

Canvas

- Log into Canvas: <http://umji.sjtu.edu.cn/canvas>
- Check the class webpage on the Canvas regularly for
 - Announcements
 - Slides
 - Grades
- Course slides will be uploaded onto Canvas before each lecture.

Getting Help

- If you have any technical questions, come to see TAs and instructor during the office hour!
 - Answering technical questions through email is inefficient.

What I Assume You Know

- Some basics of C++
 - Variables
 - Built-in data types, e.g., int, double, etc.
 - Operators, e.g., +, -, *, etc.
 - Flow of controls, e.g., if/else, while, for, switch/case, etc.
 - Functions; function declaration versus definition.
 - Arrays
 - Pointers
 - References
 - Struct

The Task of Programming

- Accept some specifications of the problem. (E.g., find the shortest way to go from my home to school.)
- Problem solving phase:
 - Design an algorithm (perhaps in pseudo-code/flow chart) that
 - 1) correctly satisfies the specification.
 - 2) is efficient in its usage of **space** and **time**.
- Implementation phase:
 - Implement the algorithm **correctly** and **efficiently**
 - 1) An implementation of an algorithm is correct if it behaves as the algorithm is intended for all inputs and in all situations.
Correctness is never negotiable!
 - 2) **Efficient** can mean fast, simple, and/or elegant.

Key Points of Ve280

- The focus of Ve280 is on the **implementation** part. Some **key points** we will learn include
 - Abstraction and its realization mechanism
 - Technique to increase code reuse
 - Technique to efficiently use memory
 - Elementary data structures
 - Some other essential parts of C++ programming

Abstraction

- One important concept about programming
 - Provides only those details that matter
 - Eliminates unnecessary details and reduces complexity
 - You already know one realization of abstraction: function (e.g. `exp(x)`), which is procedural abstraction
- We will talk about
 - Basics about abstraction
 - Procedure abstraction (i.e., function), in more detail
 - Data abstraction (i.e., class)
 - Basics about class: constructor, destructor, etc.
 - Abstract base class

Technique to Increase Code Reuse

- Function and class, which are basic ways to increase code reuse
- Class inheritance and virtual function
- Template and polymorphism
 - Template: write one thing, used for many different types

Technique to Efficiently Use Memory

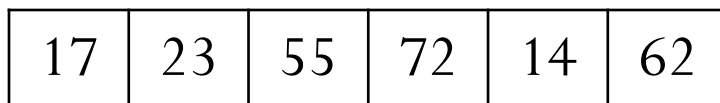
- Sometimes, the amount of memory needed to solve a problem can vary a lot
- Of course, you can write your program considering the worst-case memory usage
 - For example, a large enough array to hold data
 - However, this may lead to some waste in memory use
- We will learn a solution: **dynamic memory management**
 - Dynamic memory allocation and de-allocation

Elementary Data Structures

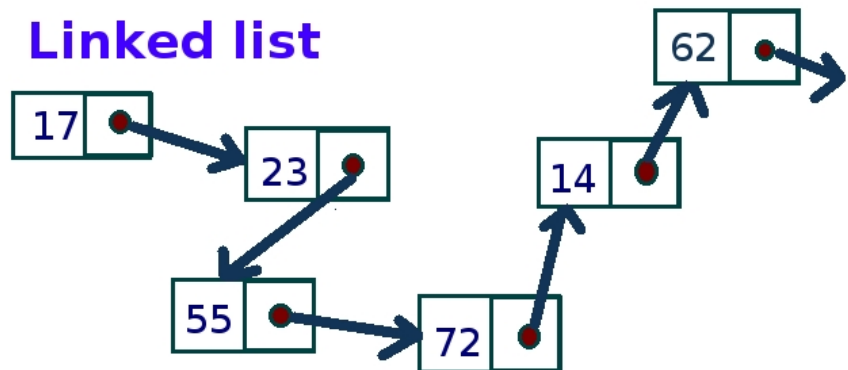
- Data structures is concerned with the **representation** and **manipulation** of data.
- All programs manipulate data.
- So, all programs represent data in some way.

Example: Store a list of numbers

Array



Linked list



Elementary Data Structures

- We will learn
 - Linked list
 - Linear list
 - Stack
 - Queue
 - Tree
- **Note**: This course only shows a few elementary data structures
 - More data structures will be taught in a following course, Ve281 Data Structures and Algorithms

Other Essential Parts

- Writing programs that take arguments
- I/O streams, including file I/O
- Error handling
- Testing
- Linux
- Bash/Perl scripting (if time permits)

Good Programming Style

Comments

**Meaningful
Naming**

Indentation

Consistency!

```
// Evaluate the polynomial on x
int poly_eval(int x, int *coef, unsigned degree) {
    int result = 0;
    int x_power = 1;
    for(unsigned i = 0; i <= degree; i++)
        result += coef[i] * x_power;
    x_power *= x;
}
return result;
}
```

In Contrast, Bad Style ...

```
int f(int a, int *b, unsigned c)
{
    int s = 0;    int p = 1;
    for(unsigned i = 0; i <= c; i++) {
        s = s + b[i] * p;
        p = p * x; }
    return s; }
```

Relation with Other Courses

- Vg101 Introduction to Computers and Programming
 - Very basic programming skills.
 - Ve280 will go in depth. To connect, we will review some basics.
- Ve281 Data Structures and Algorithms
 - Focus on the efficiency of the algorithms.
 - Ve280 focuses on correctness. It will show you some very basic data structures.

Questions?

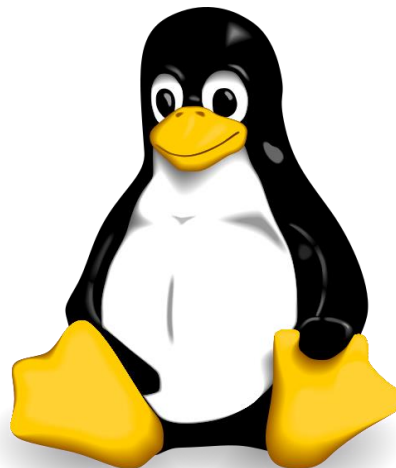
Linux

Unix

- An operating system supporting multitasking and multi-user
- Developed in 1969 by Ken Thompson, Dennis Ritchie, etc. from AT&T Bell Lab
- Many variants (Unix-like OS)
 - Linux
 - BSD (from UC Berkeley)
 - Solaris (from Sun Microsystems)
 - Android (from Google)
 - iOS (from Apple)
 - ...

Linux

- A free and open source Unix-like operating system
- First released in 1991 by Linus Torvalds
- Many distributions
 - Gentoo
 - Red Hat
 - Ubuntu
 - ...

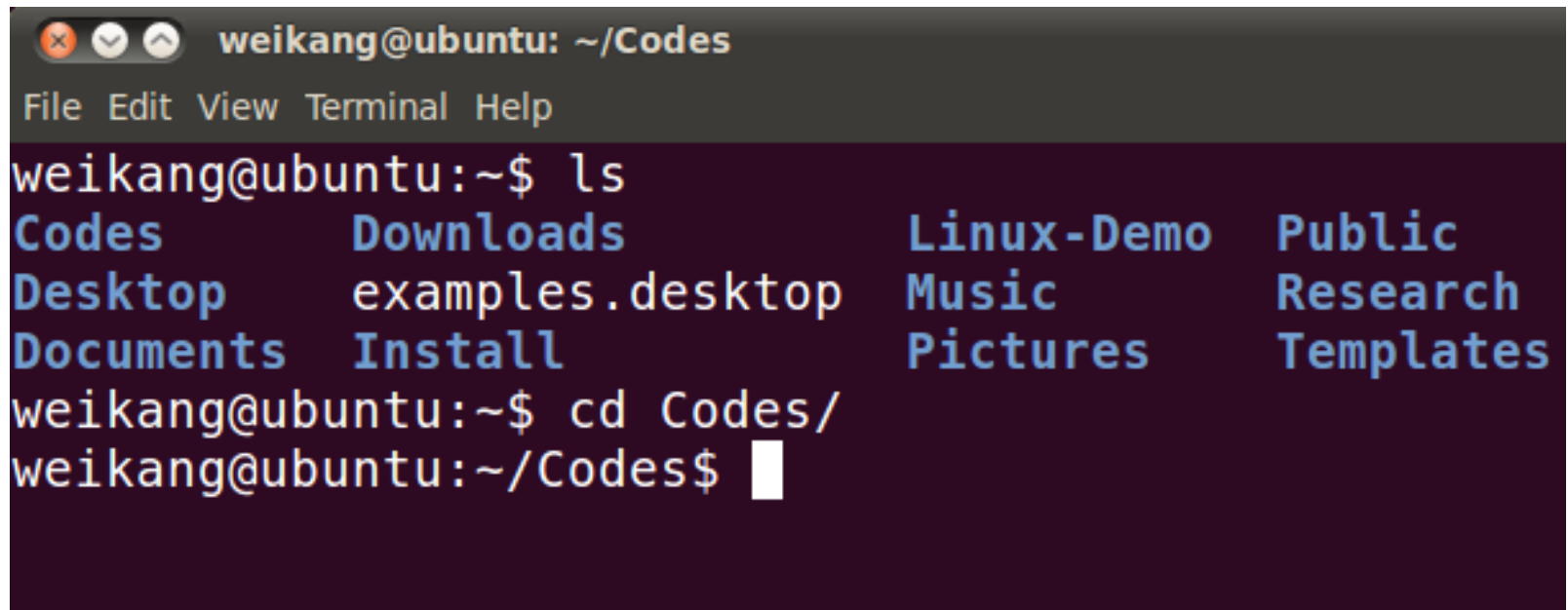


Installing Linux

- Recommended version: **Unbutu**
 - You can get the .iso file from:
<http://www.ubuntu.com/download/desktop>
 - Suggest to use the latest version.
- Install it directly on your machine
- OR Install it on a virtual machine on your Windows/Mac operating system.
 - Install a virtual machine such as VirtualBox
(<https://www.virtualbox.org/>) or VMware Player
(www.vmware.com/) first

Using Terminal in Linux

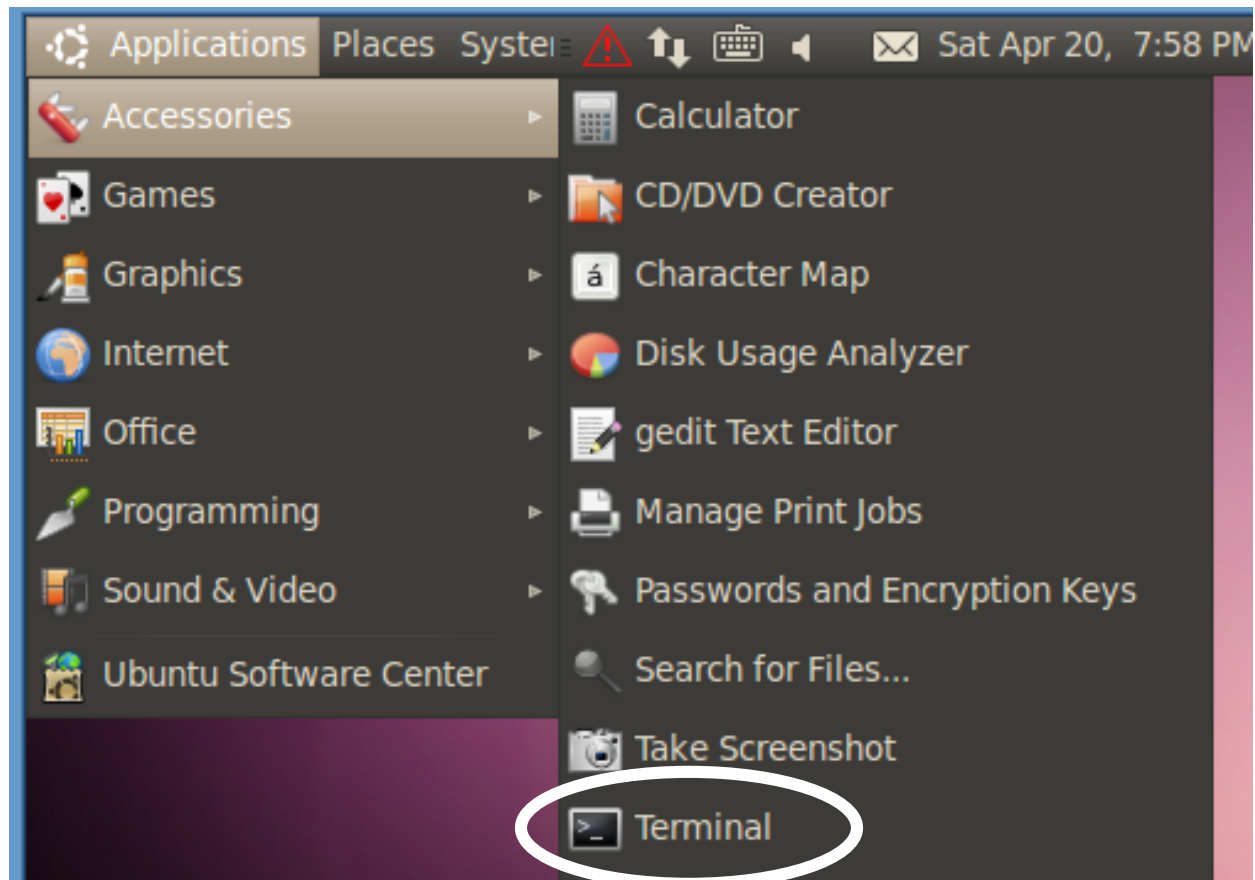
- We type commands in the terminal in Linux



The screenshot shows a terminal window titled "weikang@ubuntu: ~/Codes". The window has a menu bar with "File", "Edit", "View", "Terminal", and "Help". The terminal content shows the user running the command "ls" in the home directory, which lists various folders and files. Then, the user runs "cd Codes/" to navigate into the "Codes" directory, and the prompt changes to "weikang@ubuntu:~/Codes\$".

```
weikang@ubuntu: ~/Codes
File Edit View Terminal Help
weikang@ubuntu:~$ ls
Codes          Downloads      Linux-Demo    Public
Desktop        examples.desktop Music          Research
Documents      Install       Pictures      Templates
weikang@ubuntu:~$ cd Codes/
weikang@ubuntu:~/Codes$
```

Start a Terminal

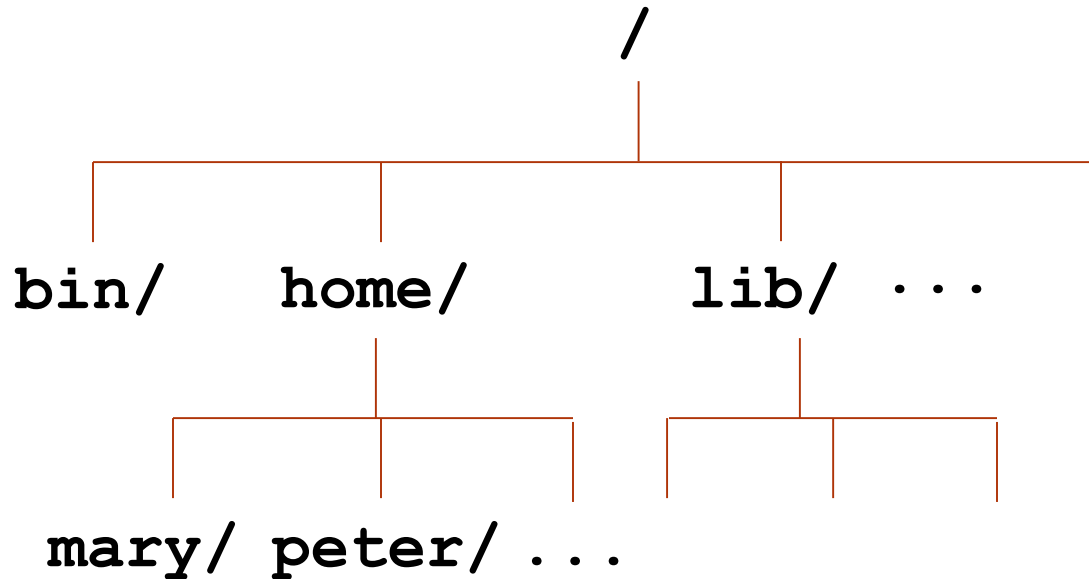


Change Directory

- Basic command: `cd pathname`
 - E.g., `cd /usr/bin`
typical path name format
- Special characters for directories
 - root directory: `/`
 - home directory: `~`
 - Linux is a multi-user operating system. It is the “home directory” of you.
 - current directory: `.`
 - parent directory: `..`

Aside: Root Directory

- Directory in Linux is organized as a tree
- The topmost directory is root directory “/”



List Contents of a Directory

- Basic command: `ls directory`
 - E.g., `ls /home`
- `ls` (i.e., “`ls`” alone): list the current working directory



Options

- `ls -l [directory]`: list in long format
- `ls -a [directory]`: list all files include the hidden files
 - Hidden files: file name begin with a dot. E.g., “.bash_history”
- In Linux, options can be combined together.
 - “`ls -la`” or “`ls -l -a`”

Aside: Long Format of File Information

- `ls -l`

group modification time

-rw-----	1	john	john	576	Apr 17 1998	weather.txt
drwxr-xr-x	6	john	john	1024	Oct 9 1999	web_page
-rw-rw-r--	1	john	john	276480	Feb 11 20:41	web_site.tar
-rwx-----	1	john	john	5743	Dec 16 1998	my_app

permission

owner

file size
(in bytes)

file name

- File permission
 - First character: '-' regular file; 'd' directory
 - Next three: read, write, execution permission of the owner
 - Next three: read, write, execution permission of the group
 - Final three: read, write, execution permission of everyone else