CS 33: Introduction to Computer Organization

Week 1   TA: Zijun Xue

LA: Kristie Lim

Get familiar with each other

- Myself
  - Zijun Xue
  - Ph.D. in ScAI(Scalable Analytics Institute) Lab
  - Doing research about Natural Language Processing/Machine Learning

  - Some interesting things about the research:
  - Deep Learning: Computational Linguistics
  - Reinforcement Learning: Conversational AI and AlphaGo

Target

- Help everyone have a fun and effective learning experience
  - Ask questions
  - Work on the labs/homework/puzzles
- Make some good friends in this class
  - Help each other

General Information

- Contact Information
  - Email: xuezijun@ucla.edu
  - Office Hour: Wed 3:00-5:00PM, BH 3256-S
- Course Website
  - https://ccle.ucla.edu/course/view/19S-COMSCI33-1?section=0

General Information

- Where to Ask Question
  - General question: Campuswire
  - Personal question: Office Hour or Email
- Grading Breakdown
  - 40%	labs (4, each 10%)
  - 5%	homework (5, each 1%)
  - 55%	exams

General Information

- The content of discussion sessions usually are about
  - Explain about Labs and homework
  - Review of important contents in lecture
  - Some exercises
- Tips for this class
  - Make good use of the textbook
  - Think more, ask questions
  - Focus on the labs

Outline

- Introduction to Linux Command
- Lab 1: Data Lab
- Lecture overview
  - Binary Representation
  - Bit operator
  - Logical operator
  - Shift operators
- Appendix
  - C language
  - SEASNET Server Login

How to start in Linux(a super fast class)

- Linux commands have the following format
  - <command name> <option> <argument>
  - Options and arguments are optional for some commands
  - Options specify the behavior of the command

Move around in linux

- pwd: print working directory
- ls: list contents of current directory
  - Try the options: ls –l, ls –a, ls -al
- cd: change directory
  - ~ home directory
  - . current directory
  - .. parent directory
  - / root directory

Pick and drop the file in linux

- mkdir
  - creates a new directory
  - e.g. mkdir cs33
- mv
  - move or rename file or directory
  - mv <option> <source> <destination>
- cp
  - copy file or directory
  - -r (recursive flag, usually used to copy a directory)
  - cp <option> <source> <destination>

Edit in linux

- Editing files in linux: use **vi** or **emacs**
  - vi abc.txt
  - emacs abc.txt
- Options in vi
  - :w – save
  - :q – quit
  - :q! – quit and not save
  - :wq – quit and save

Compile in Linux

- Compile programs with gcc
  - gcc main.c
    compile a source file into executable file, default name: a.out
  - gcc main.c –o main
    compile a source file into executable file named main
  - gcc main.c –O2 main
    compile a source file with optimization level 2
  - Execute a file:
    ./main

option	optimization level	execution time	code size	memory usage	compile time
-O0	optimization for compilation time (default)	+	+	-	-
-O1 or -O	optimization for code size and execution time	-	-	+	+
-O2	optimization more for code size and execution time	--		+	++
-O3	optimization more for code size and execution time	---		+	+++
-Os	optimization for code size		--		++
-Ofast	O3 with fast none accurate math calculations	---		+	+++

Fall in love in linux

- If you have questions about Linux commands, ask **man**
- man whoami
- man git
- man vim
- man cp
- man ls
- man cd
- man man

Lab 1: Data lab

- Setting up the environment
  - Please use the SEASnet server: lnxsrv09.seas.ucla.edu
  - The version of gcc you need to use is 5.2.0
  - Check: **gcc -version** This version of gcc in this directory /usr/local/cs/bin of lnxsrv09.seas.ucla.edu
- If 5.2.0 is not the version that you see, add the above path into environment variables
- Environment variables are variables that are known by the operating system or by an instance of a terminal.
  - PATH=$PATH:/usr/local/cs/bin
  - Echo $PATH

Lab 1: Data lab

- "echo $0" - Will tell you your current shell.
  - ex. -tcsh
- "echo $SHELL" - Will tell you your default shell.
  - ex. /usr/local/bin/tcsh

```
Mingdas-MacBook-Pro:~ MingdaLi$ ps -p $$
  PID TTY           TIME CMD
20803 ttys001    0:00.61 -bash
Mingdas-MacBook-Pro:~ MingdaLi$ echo $0
-bash
Mingdas-MacBook-Pro:~ MingdaLi$
```

- First, determine what shell you're using, Type commands:
  - echo $0
  - ps -p $$
- Command 1
  - print out the type of shell
- Command 2
  - print out a table with a single row whose "CMD" field should match the output of echo $0
  - If the results of these commands differ, you may have some script that is executing another shell upon logging in.

Lab 1: Data lab

- The bash shell
  - Open the file ".bash_profile" from a text editor in your home directory
  - Add the line: "export PATH=/usr/local/cs/bin:$PATH"
  - Restart

csh or tcsh, in your home directory, do the following:

- Open the file ".login" from a text editor.
- Add the line: "set path=(/usr/local/cs/bin $path)"
- Restart

Lab 1: Data lab

- Setting up the datalab
  - Download datalab.tgz from CCLE.
  - Copy it to a directory of your choosing
  - .tgz is a compressed file type
  - Uncompress it with the command:

    tar -zxvf datalab.tgz

Lab 1: Data lab

- Running the datalab
  - Compiling: “make”
  - Testing your code (correctness): “./btest”
  - Testing your code (follows rules): “./dlc -e bits.c”
- See INSTRUCTIONS.txt and README for more

Lecture Review: binary representation

- What can binary integers be used to represent?
  - Signed, Unsigned Numbers
  - Bool value (true/false)
  - Sets of integers
- Different bases
  - Decimal: Base 10
  - Binary: Base 2
  - Hexadecimal: Base 16
  - Convert: Decimal -> Binary e.g. 2017_{10}
  - Convert: Binary -> Hexadecimal e.g. 11111100001_2

Binary Number Representation

- 0110 0101
- Base 2 number representation.
- Each digit can only be one of two options, 0 or 1, hence, “bi”-nary.

Unsigned Binary Representation:

- The base 2 method of representing numbers (compare against decimal representation, which is base 10)
- Consider the decimal number 2340:
  - $2340 = 2*10^3 + 3*10^2 + 4*10^1 + 0*10^0 = 2340$
- Consider the binary number 1010:
  - $1010 = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 10$
- An N bit binary number has 2^N values or a range of $[0, 2^N-1]$

Decimal to Binary (informal)

- Let d be a decimal value.
- We build the binary number from the least significant bit to the most significant bit. Start from the least significant bit.
- To get the current bit of the number, take the modulo of d and 2 (I.e. d % 2).
- Then, integer divide d by 2, that is to say if d were 5, 5 integer divided by 2 would be 2. Now we are dealing with the next bit
- Repeat the process until d is 0.

Decimal to Binary Example

1. D = 23

   b0 = 23 mod 2 = 1
2. D = 23 / 2 = 11

   b1 = 11 mod 2 = 1
3. D = 11 / 2 = 5

   b2 = 5 mod 2 = 1
4. D = 5 / 2 = 2

   b3 = 2 mod 2 = 0
5. D = 2 / 2 = 1

   b4 = 1 mod 2 = 1
6. D = 1 / 2 = 0

b = 10111

Number representations

- One binary digit is a bit. Can be one of 2 values.
- x bits has a range of 2^x values.
- 8 bits = byte. Has a range of 256 values.
- To have a practical range of memory, we need a lot of address space, which means a lot of bits.
- In modern computers, that's likely going to be 32 or 64 bits.
- Scientifically speaking, this is known as “a lot”. This is, in part, why we have:

bit and byte

- Bit: smallest unit of data
- 1 byte = 8 bits
- A bit has one of two values: 0 or 1

C Data Type	Size (bytes)
char	1
short	2
Int	4
Long	8

Hexadecimal Representation

- The base 16 method of representing numbers.
- Hexadecimal digits range from 0-9 and A-F where A-F correspond to values 10-15
- The prefix “0x” is used to denote numbers as written in hexadecimal.
- In hexadecimal:
  - $0x234C = 2*16^3 + 3*16^2 + 4*16^1 + 12*16^0 = 9036$

Signed Binary: Two's Complement

- How do we represent negative numbers?
- The two's complement of a number is technically it's value subtracted from 2^N.
- In two's complement, most bits have the same contribution as in unsigned. The value of the i-th bit is 2^i (assuming i starts from 0).
- However, the most significant bit of an N bit number has a value of -2^{N-1} instead of 2^{N-1}

Signed Binary: Example

- Assume we're dealing with four bit numbers.
- Consider the unsigned binary number 1010:
  - $1010 = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 10$
- Now consider the signed binary number 1010:
  - $1010 = 1*(-(2^3)) + 0*2^2 + 1*2^1 + 0*2^0 = -6$

Signed Binary: Notes

- The value of a signed binary number depends on the number of bits there are.
  - Four bit signed: 1111 = -1
  - Five bit signed: 01111 = 15
- An N-bit signed binary number has 2^N possible values with a range of $[-2^{N-1}, 2^{N-1}-1]$.
- REMEMBER THIS: The range of a twos complement signed binary number is not symmetrical around 0.
- Henceforth all signed binary is two's complement unless otherwise specified.

Bitwise Operators

- Bitwise operators operate on a single bit
- AND : &
  - 1 if both inputs are 1, 0 otherwise
- OR : |
  - 1 if at least one input is 1, 0 otherwise
- XOR : ^
  - 1 if two inputs have different values, 0 otherwise
- NOT : ~
  - 1 if the input was 0, 0 otherwise

Bitwise Operators Exercise

- 11110011 & 10101010
- 11110011 | 10101010
- 11110011 ^ 10101010
- ~11110011

Logical Operators

- Difference
  - Bitwise op: operate on each individual bit of a number
  - Logical op: operate on the number as a whole
- Operation: Non-zero numbers are interpreted as 1 and 0 is interpreted as... 0
- Operators: &&, ||, !
  - 1011 && 1100 <=> 1, 1011 && 0 <=> 0
  - 1011 || 0 <=> 1
  - ! 1011 <=> 0

Logical Operators Exercise

- 11110011 && 10101010
- 11110011 && 00000000
- 11110011 || 10101010
- 11110011 || 00000000
- !11110011
- !00000000

Useful Tricks: Division via shifting

- Consider 4-bit unsigned:
  - 1100 = 12
  - 1100 >> 1 = 0110 = 6 (looks good...)
- Consider 4-bit signed:
  - 1100 = -4
  - 1100 >> 1 = 0110 = 6 (hrm?)

Useful Tricks: Division via shifting

- Previously, we tried logical right shifting (shift in zeros, but that didn't seem to pan out). This is where arithmetic right shifting steps in.
- Consider 4-bit signed:
  - 1100 = -4
  - 1100 >> 1 = 1110 = -2
- Logical shifting maintains correct values for unsigned operations while arithmetic shifting maintains correct values for signed operations.

Shift Operators

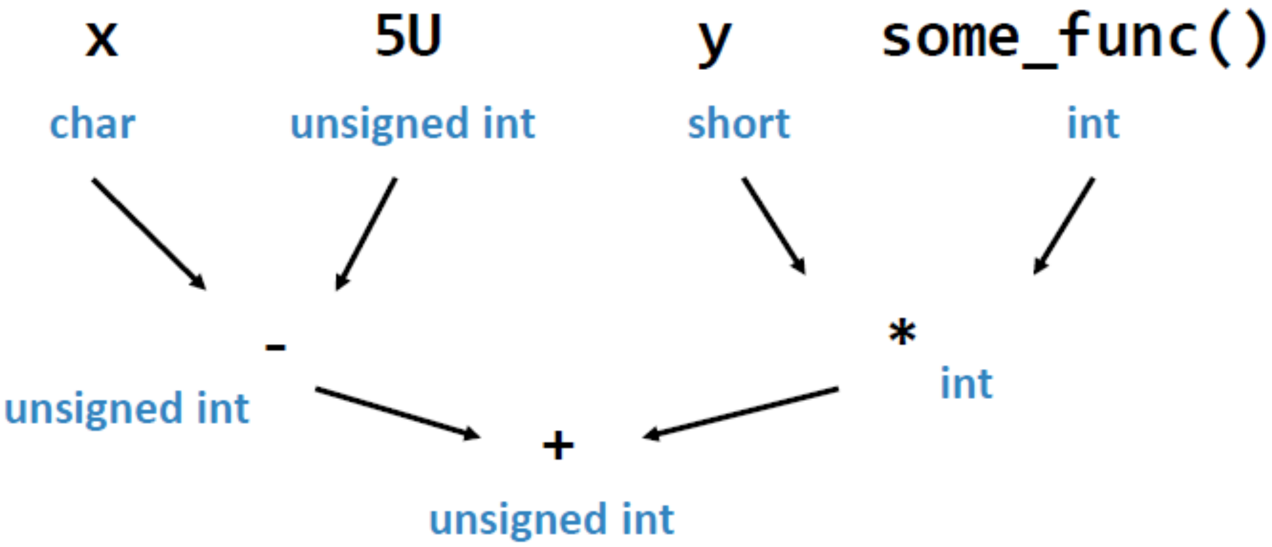
- Left shift : << The same
  - 0111 << 1 = 1110
- Right shift : >>
  - 1011 >> 1 = 0101 (logical)
  - 1011 >> 1 = 1101 (arithmetic) Keep the sign bit 1 and fill with 1.

C Type Casting

```
char x;
short y;
int some_func();

… = (x - 5U) + (y * some_func);
```

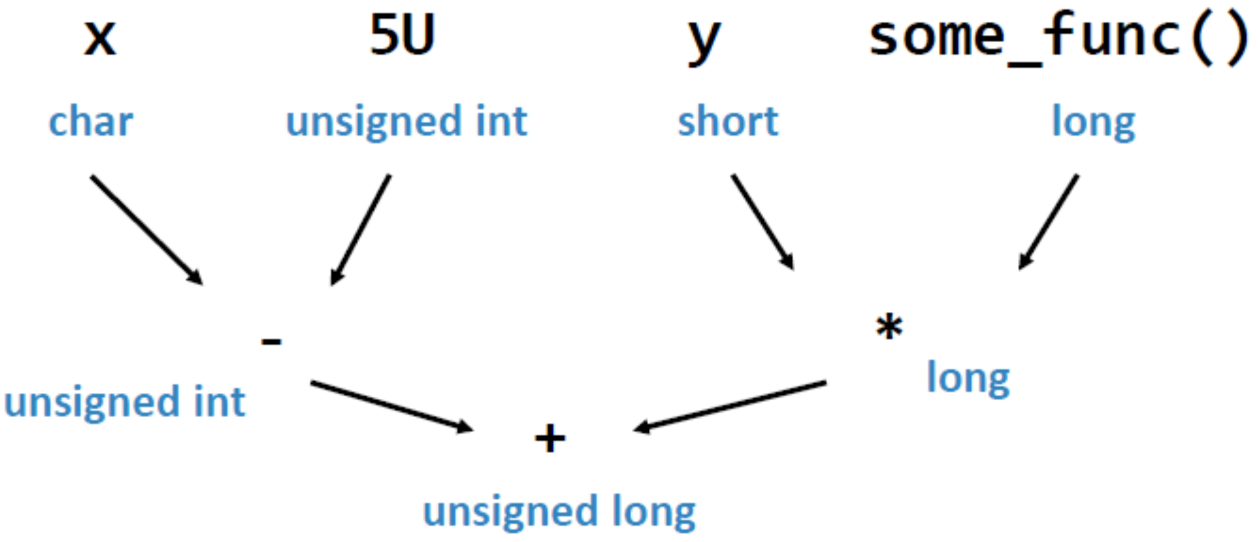
C Data Type	Size (bytes)
char	1
short	2
Int	4
Long	8



C Type Casting

```
char x;
short y;
int some_func();

… = (x - 5U) + (y * (long)some_func);
```



Seasnet

www.seasnet.ucla.edu/secure-remote-login-file-transfer/

Secure Remote Login File Transfer

For secure remote login and
file transfer, use ssh and sftp (instead of telnet and ftp).

To run graphical
application on a remote unix server, see X11 Forwarding.

Windows Clients

- PuTTY SSH
  - How to install
  - How to use
- WinSCP freeware SFTP and SCP client for Windows
- X11 Forwarding
- Xming X Server for Windows

Unix Clients

- Example: how to use ssh
- Example: how to use sftp

Macintosh Clients

- Note that Mac OS X includes OpenSSH by default.
- OpenSSH Mac OS clients

Why seasnet?

- Linux server
- All your homework will be texted on Seasnet.

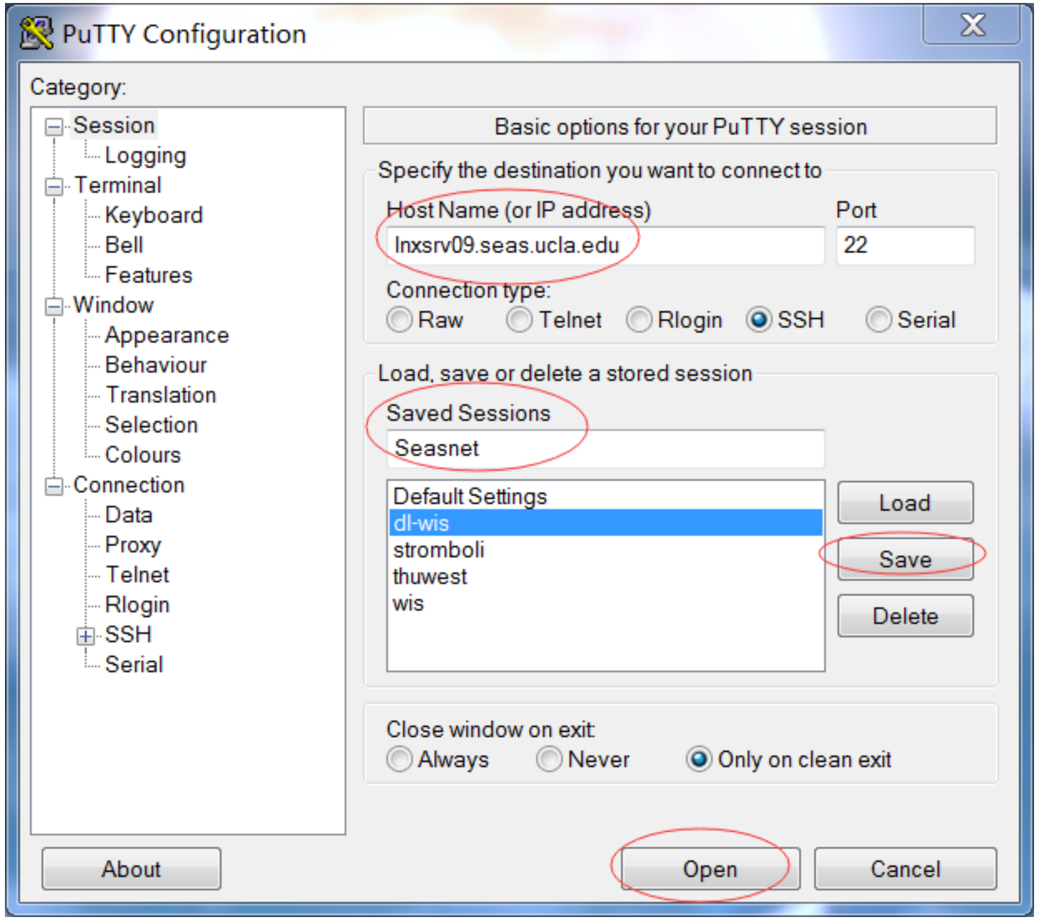
- What you need to do on seasnet?
  - Login
  - File transfer
  - Editing/Playing/Learning/(Suffering) in Linux

Login

- How to log in?
  - First, Get a Seasnet account ☺(BH second floor)
- Where to log in?
  - If you are a Linux/Mac user, use ssh.
    - ssh [username]@lnxsrv07.seas.ucla.edu
    - ssh zijun@lnxsrv.seas.ucla.edu
  - If you are a windows user, use Putty(next slide)

Seasnet Server Login: Putty

- First Run
  - Type in the host name
  - Type a name for saved sessions
  - Click “save”
  - Click “open”
  - Type in your user name and password
- For the next time, just double click the saved session



File transfer

- Windows user
  - WinSCP: An scp client
  - Download link: https://winscp.net/download/WinSCP-5.9.4-Portable.zip
  - Usage: similar to Putty, need to type in the server address and then user name and password
- Mac/Linux user
  - Use scp command
    - scp yourSeasUsername@lnxsrv.seas.ucla.edu:source destination
    - Example: scp zijun@lnxsrv.seas.ucla.edu:~/CS33/text.txt .
  - Use cyberduck(next slide)

File transfer

- Copying files between local machine and server: mac OS
  - Cyberduck: a GUI client designed for mac OS
  - Download: https://update.cyberduck.io/Cyberduck-5.4.0.23761.zip
  - Usage:
    - Click “open connection”
    - Select “SFTP”
    - Type in the address of server
    - Type in user name and password
    - Click “Connect”

C Language As opposed to C++

- In a (very simplified) nutshell, C++ is an extension to C.
- The syntax of the language is nearly identical, but you will find that C lacks certain features, namely the “Object Oriented” paradigm.
- Some features are analogous, but have different names.

C (as opposed to C++)

- In C++:
- for(int i = 0; i < size; i++) ...
- By default, gcc uses a 1990's C standard which prohibits declarations in “for” loops. As a result, you will have to do either
- int i;
- for(i = 0; I < size; i++)
- Or explicitly use gcc to compile with a different C standard:
- gcc -std=c99 temp.c

C Language: Struct

- No classes in C
- Function: package related data (variables of different types) together
- Single name is convenient (rename: *typedef*)

```
struct Student {
        char name[64];
        char UID[10];
        int age;
        int year;
};
struct Student s;
```

```
typedef struct {
        char name[64];
        char UID[10];
        int age;
        int year;
} Student;
Student s;
```

C Language: C Struct vs. C++ Class

- C structs cannot have member functions
- There's no such thing as access specifiers in C
- C structs don't have constructors defined for them

- C++ classes can have member functions
- C++ class members have access specifiers and are private by default
- C++ classes must have at least a default constructor

C language: memory allocation

- In C, these declarations force you to be more specific.
- Instead of “new”, use “malloc” and instead of “delete”, use “free”. e.g.

```
char * c_arr = (char *) malloc(sizeof(char) * 10);
free(c_arr);
```

- void * malloc(size_t size) : allocate some amount of memory.
  - size: the number of bytes to allocate
- void * calloc(size_t num, size_t size) : allocate some amount of memory and initialize it as 0
- size: size of each element
  - num: the number of elements
- void * realloc(void * ptr, size_t size) : takes an existing pointer and reallocates the size memory allocated by that pointer, changing it's location if necessary
  - size: new number of bytes to allocate

C language: Pointers Review

- Pointers: Variables that store **memory addresses**
- Declaration
  - <variable_type> *<name>;
  - int *ptr; //declare ptr as a pointer to int
  - int var = 77; // define an int variable
  - ptr = &var; // let ptr point to the variable var
  - void* pointer: a pointer whose value can be with any data type

C language: Dereferencing Pointers

- Accessing the value that the pointer points to
- Example
  - double x, *ptr;
  - ptr = **&**x; // let ptr point to x
  - *****ptr = 7.8; // assign the value 7.8 to x