Introduction to Computer Organization

DIS 1H – WEEK 1

DIS 1H

TA: Shikhar Malhotra

- Email: shikharmalhotra1@gmail.com
- Office Hours: M 4:00 P.M – 6:00 P.M.
- TA Room: BH 2432

Today's Schedule

- Administrative information
- Lecture Review
- Linux overview and accessing the SEASnet Linux servers
- C (as opposed to C++)
- Homework

Administrative information

- Discussion Slides will be e-mailed to the group
- Labs carry 40% of the total grade
- You are encouraged to use Piazza for any doubts

Getting Started

- This class is based around C, not C++

- You are highly recommended to ditch Visual Studio and work in a Linux environment, specifically the SEASnet Linux servers.

- Your assignments will be tested on the SEASnet Linux servers

Setup

In order to login to SEASnet, you need to be connected to wireless networks on campus

OR

login with VPN software

https://www.it.ucla.edu/bol/services/virtual-private-network-vpn-clients

Setup Guide for Windows

Download PuTTY

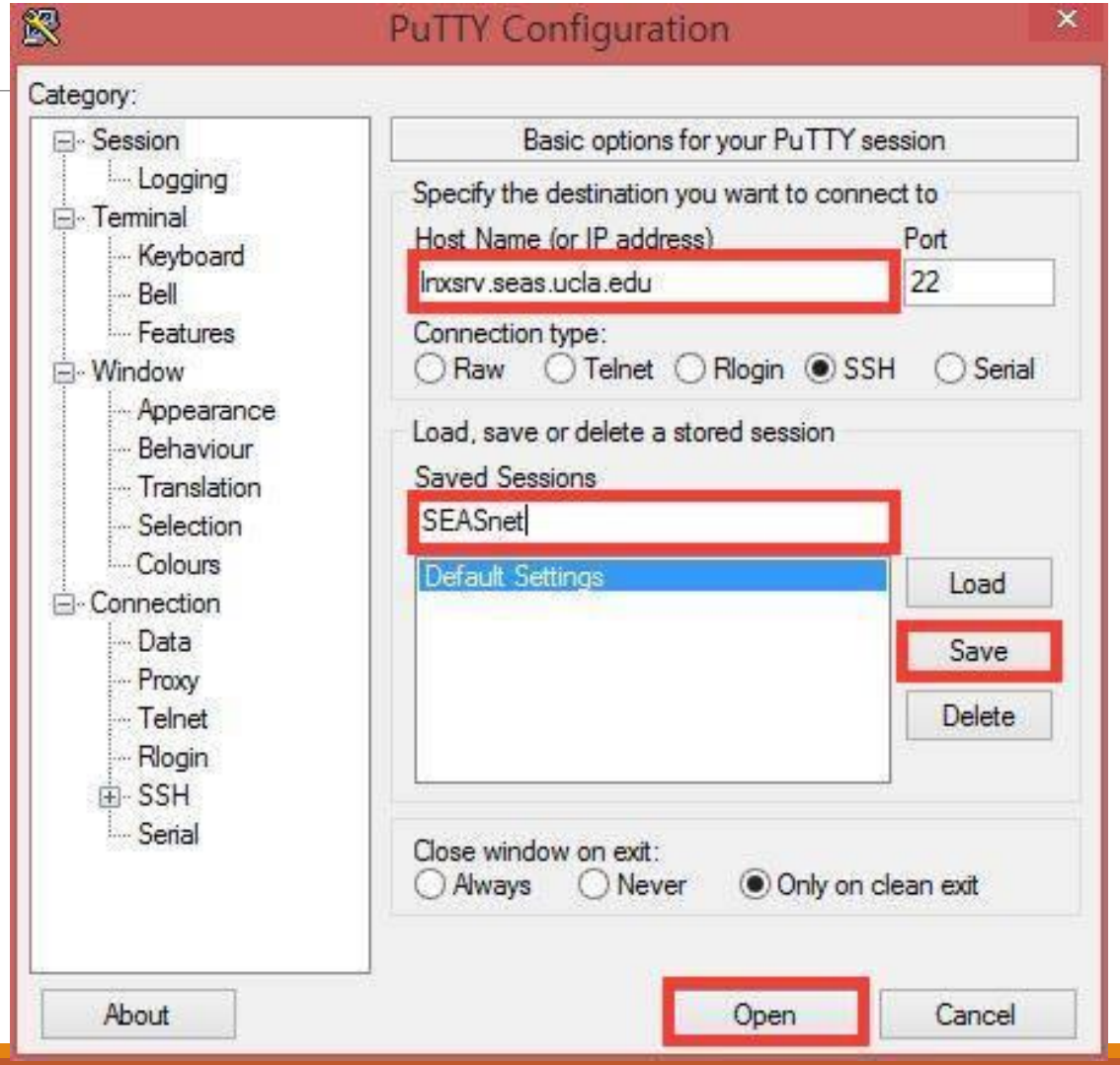
- https://the.earth.li/~sgtatham/putty/latest/x86/putty.exe

PuTTY

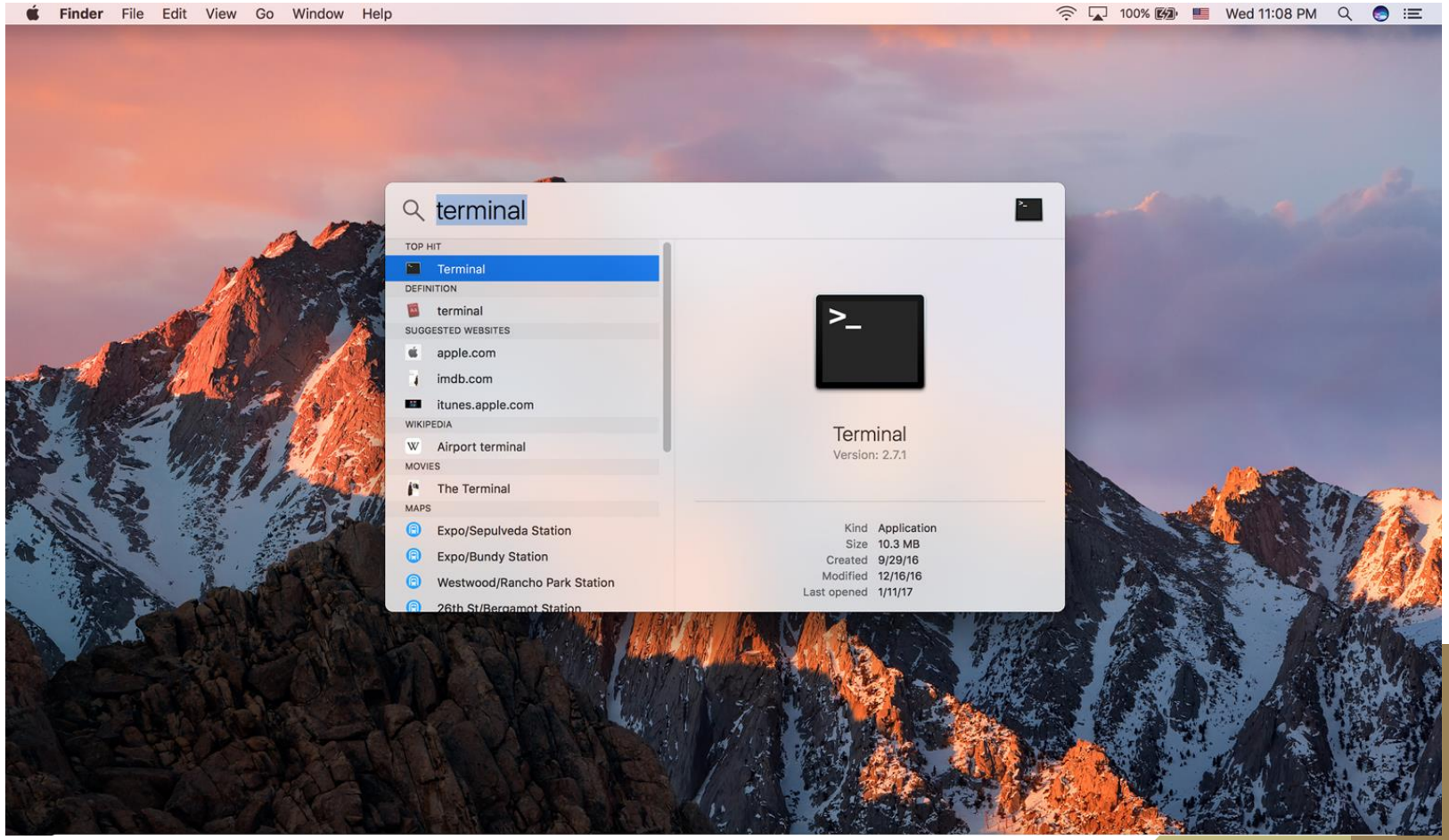
First Run

- Type *lnxsrv@seas.ucla.edu* for Host Name
- Type SEASnet for Saved Sessions
- Click Save
- Click Open
- Type your SEASnet username and password

Double-click SEASnet under Saved Sessions in the future



Setup Guide for Mac



Terminal

Type `ssh <SEASnet username>@lnxsrv.seas.ucla.edu`

Enter your password

Copying between local machine and SEASnet

- For Windows users:

– WinSCP: An scp client

– scp stands for secure copy. You use it to secure copy.

Download WinSCP

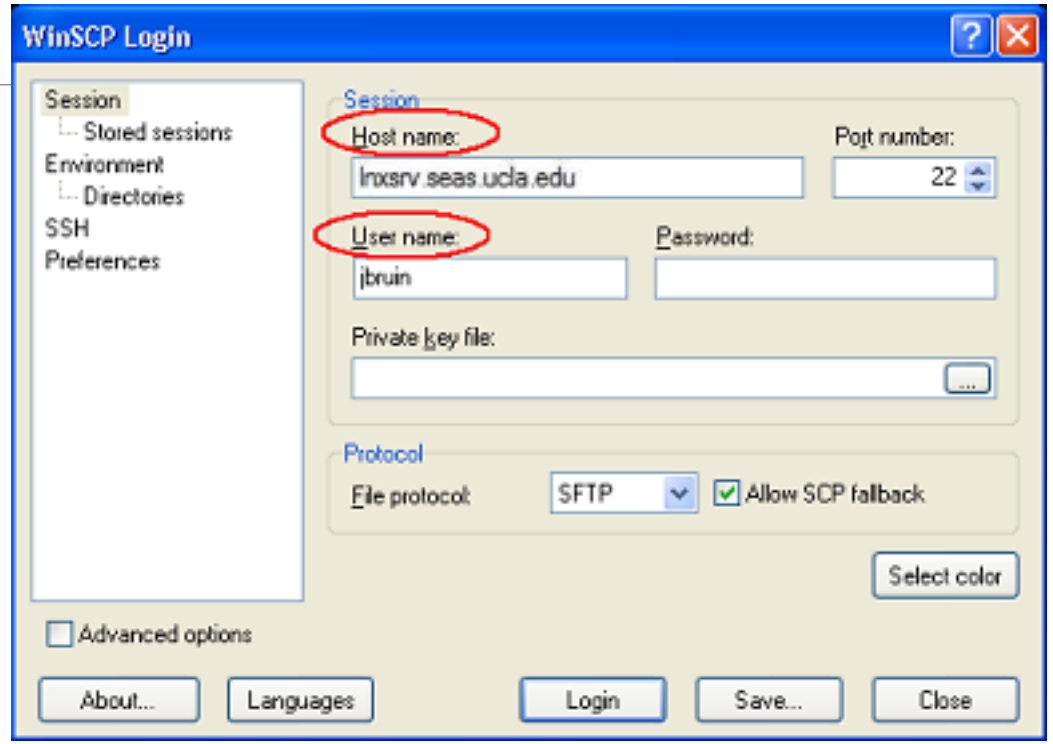
  - http://winscp.net/download/winscp427setup.exe

WinSCP

Type *lnxsrv.seas.ucla.edu* for Host name

Type your SEASnet username and password

Right click a file and select upload or download



Copying between local machine and SEASnet

Download Cyberduck

- https://update.cyberduck.io/Cyberduck-5.2.2.21483.zip

Cyberduck

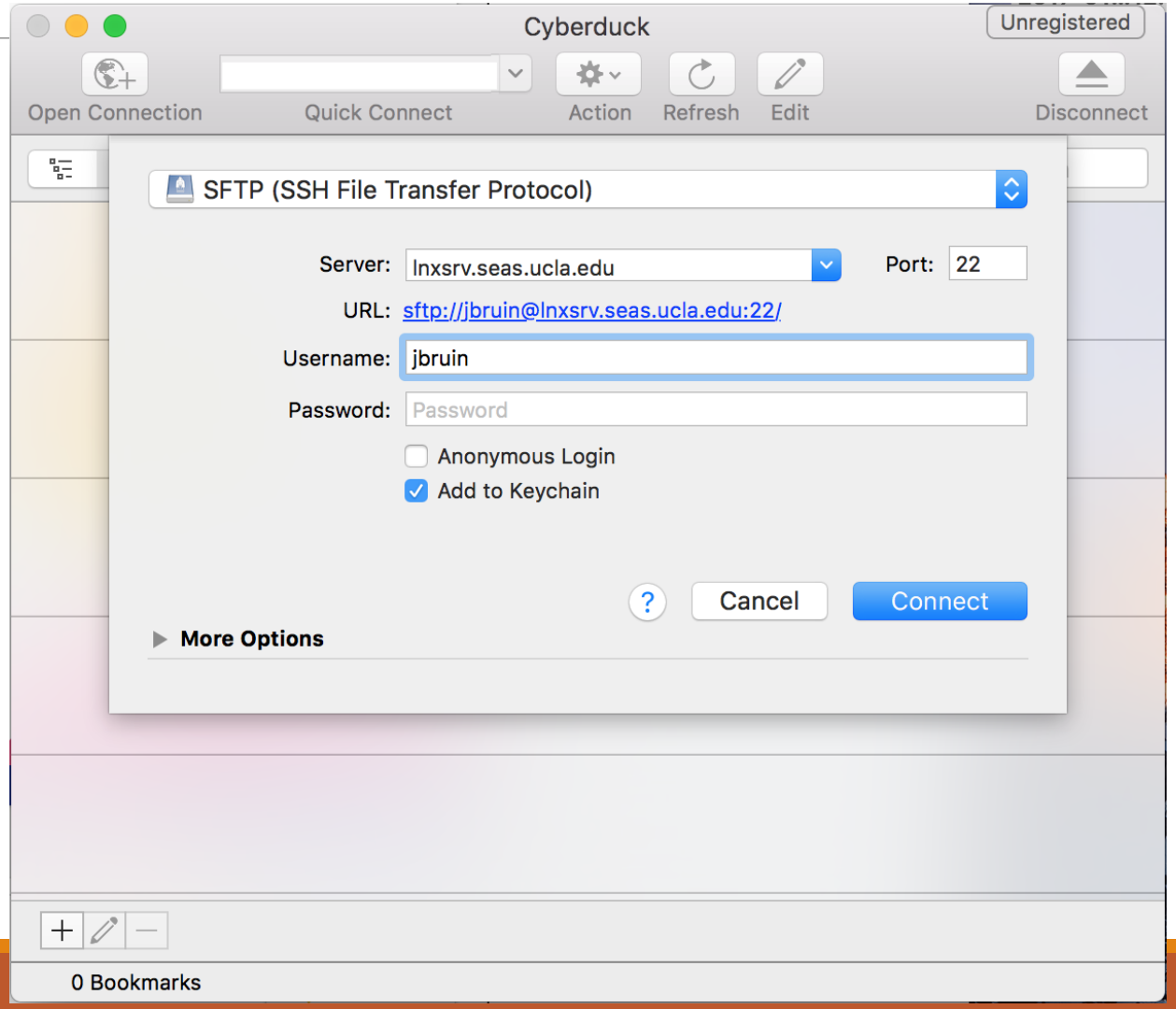
Click Open Connection

Select SFTP (SSH File Transfer Protocol)

Type *lnxsrv.seas.ucla.edu* for Server

Type your username and password

Click Connect



Linux Introduction

- Linux is the operating system/kernel that you will be learning about in this

- The primary interface into Linux that you'll be using is a command line interface akin to MSDOS

Useful Linux Commands

- Linux command format:
- [command name] -X -Y -Z [argument]
- (X, Y, and Z are optional flags)
- Flags modify/specify the behavior of the command

Useful Linux Commands

- pwd – print working directory
- ls – list contents of current directory

– ls -l (“-l” flag will print contents in long form)

- cd – change directory

– cd a (navigates to the “a” directory which is located in the current directory)

– cd .. (navigate to the parent directory)

– cd . (navigate to the current directory)

- mkdir <dir_name> – creating a new directory

Useful Linux Commands

- Editing files. If you're interested familiarizing yourselves with Linux (which will have to happen eventually), it is recommended that you use “vi” or “emacs”.

– vi text.txt

– emacs text.txt

- Commands that start with a colon :

:w save

:x save and quit

:q quit (will trigger error if file hasn't been saved)

:q! quit without saving

Useful Linux Commands

- The standard Linux C compiler is gcc.

– gcc main.c (compile the file main.c into an executable file with default name “a.out”)

– gcc main.c -o main (compile the file main.c into an executable file called “main”)

– gcc main.c -O2 (compile the file with optimizations, level 2)

- Executing executables

– ./main (executes the executable file called “main”)

C (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 (as opposed to C++)

Dynamic memory allocation

- In C++:

– char * c_arr = new char[10];

– delete c_arr;

– “new” allows you to specify repetitions of a specific data type.

C (as opposed to C++)

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

– char * c_arr = (char *) malloc(sizeof(char) * 10);

– free(c_arr);

- Note: These are analogous but not the same.

- “malloc” and other “_alloc” variations operate on the principle that you're specifying a specific amount of memory to allocate rather than a specific data type.

C (as opposed to C++)

Instead of:

– int x = 10;

– cout << x;

● You'll use “printf”

– printf(“hello”);

– printf(“%d”, x);

● printf takes in as the first parameter a string to print out that is populated with format codes that correspond to the remaining arguments.

Homework

- *Checking the gcc version*
  - cd /usr/local/cs/bin
  - gcc –version

- *PATH=$PATH:/usr/local/cs/bin*
  - Then you can type *echo $PATH* to check whether it works.

- Work with lnxsrv07 or lnxsrv09 for latest version of gcc

Homework

- *bool* **__builtin_add_overflow** *(type1 a, type2 b, type3 *res)*

- “Promote the first two operands into infinite precision signed type and perform addition on those promoted operands. The result is then cast to the type the third pointer argument points to and stored there. If the stored result is equal to the infinite precision result, the built-in functions return false, otherwise they return true”

- Don’t use other functions like *bool* **__builtin_add_overflow_p** *(type1 a, type2 b, type3 c)* because they are supported in gcc versions >= 7.

Homework

- *bool* **__builtin_add_overflow** *(type1 a, type2 b, type3 *res)*

```
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>
int main(){
int A = INT_MAX, B = 3;
int C;
bool a = __builtin_add_overflow (A, B, &C);
printf("%s", a ? "true" : "false");
return 0;}
```

Homework

gcc -fwrapv -O2 -Wall -Wextra -S hw1/2.73.c

- -fwrapv – instructs the compiler that integer overflow wraps
- -O2 – GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff. As compared to -O, this option increases both compilation time and the performance of the generated code.
- -Wall - Turns on all optional warnings which are desirable for normal code. This enables all the warnings about constructions that some users consider questionable, and that are easy to avoid
- -Wextra - Warn if a comparison is always true or always false due to the limited range of the data type, but do not warn for constant expressions. For example, warn if an unsigned variable is compared against zero with '<' or '>=‘
- -S – Remove all symbol table and relocation information from the executable

End