ELEC / COMP 177 – Fall 2011

Computer Networking → C Programming Essentials

Some slides from Kurose and Ross, *Computer Networking*, 5th Edition

Experience

- Prior experience in programming languages
  - **C++ programming?**
  - **Java programming?**
  - **C programming?**
  - **Other languages?**
- Prior experience in Linux / FreeBSD /Unix
  - **Yes?**
  - **No?**

Tools for Homework/Projects

- My *preference* is for you to use the class server for the next homework and all projects
  - We'll discuss the key software elements today and do a quick lab
    - X-Windows, SSH, and Eclipse
- Advanced students: You can easily replicate the class server on your own laptop!
  - *If you diverge too far from the class server, make sure your project still compiles and runs on the official machine*
    - **Otherwise, zero points!**

What is X Windows?

- Network protocol for GUI displays
  - Just the raw display pixels and keyboard/mouse inputs
- "Backwards" notion of client/server
  - The client is where the GUI program runs and produces graphics information
  - The server is where the graphics information is actually displayed!
- Server is built in for Mac/Linux
  - Use "Xming" program on Windows

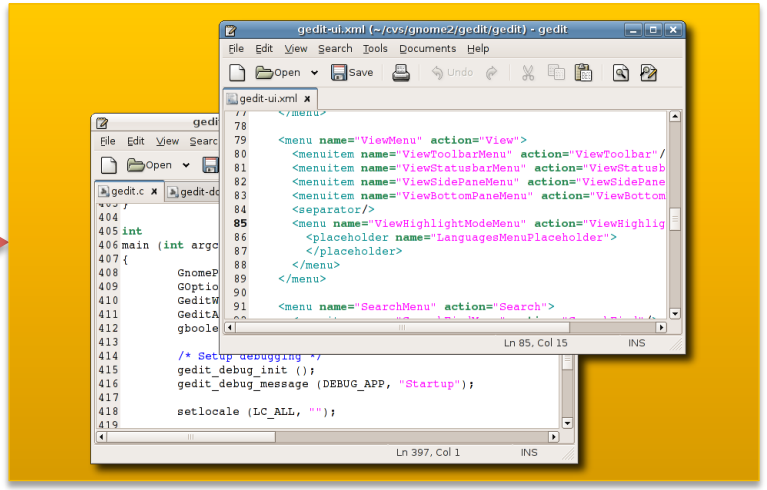
What is X Windows?

Computer 1 – "Client"

Gedit text editor

Graphics information

Computer 2 – "Server"

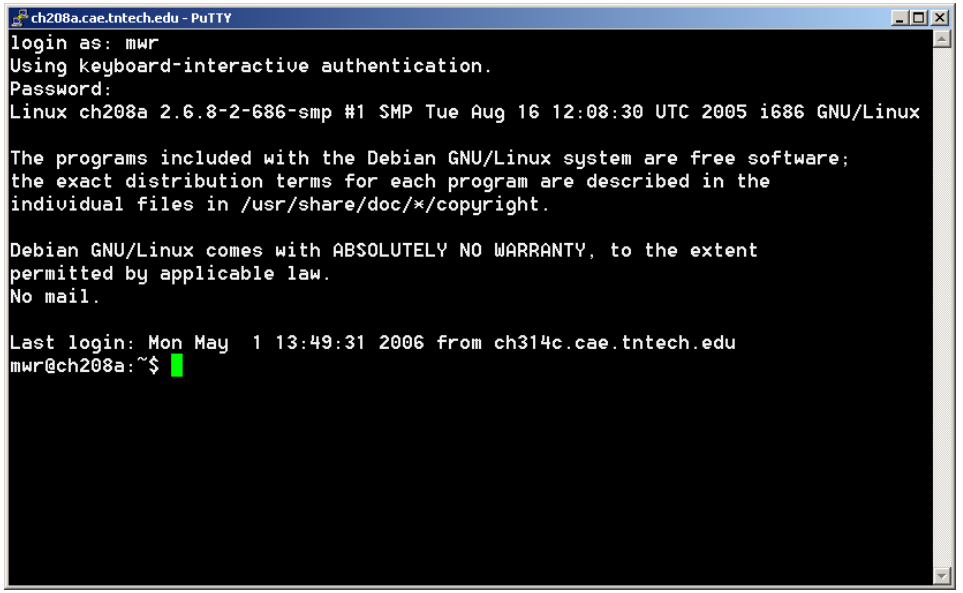


*Program **runs** on this computer!*

*Program is **displayed** on this computer!*

What is SSH?

- Telnet (command-line access)
- Encrypted/secure
- Can **tunnel X Windows displays** over SSH
- Example programs:
  - Windows:
    Putty (free)
  - Mac/Linux:
    "ssh" built in



Server

- `ecs-network.serv.pacific.edu`
  - Virtual machine
    - Sharing a big 8-core, 64GB RAM server in the campus datacenter
  - Linux / Ubuntu 10.10 64-bit
- Network – behind the campus firewall
  - HTTP (web) access: Worldwide
  - SSH access: On-campus only
    - **So how can you do your homework/projects?**

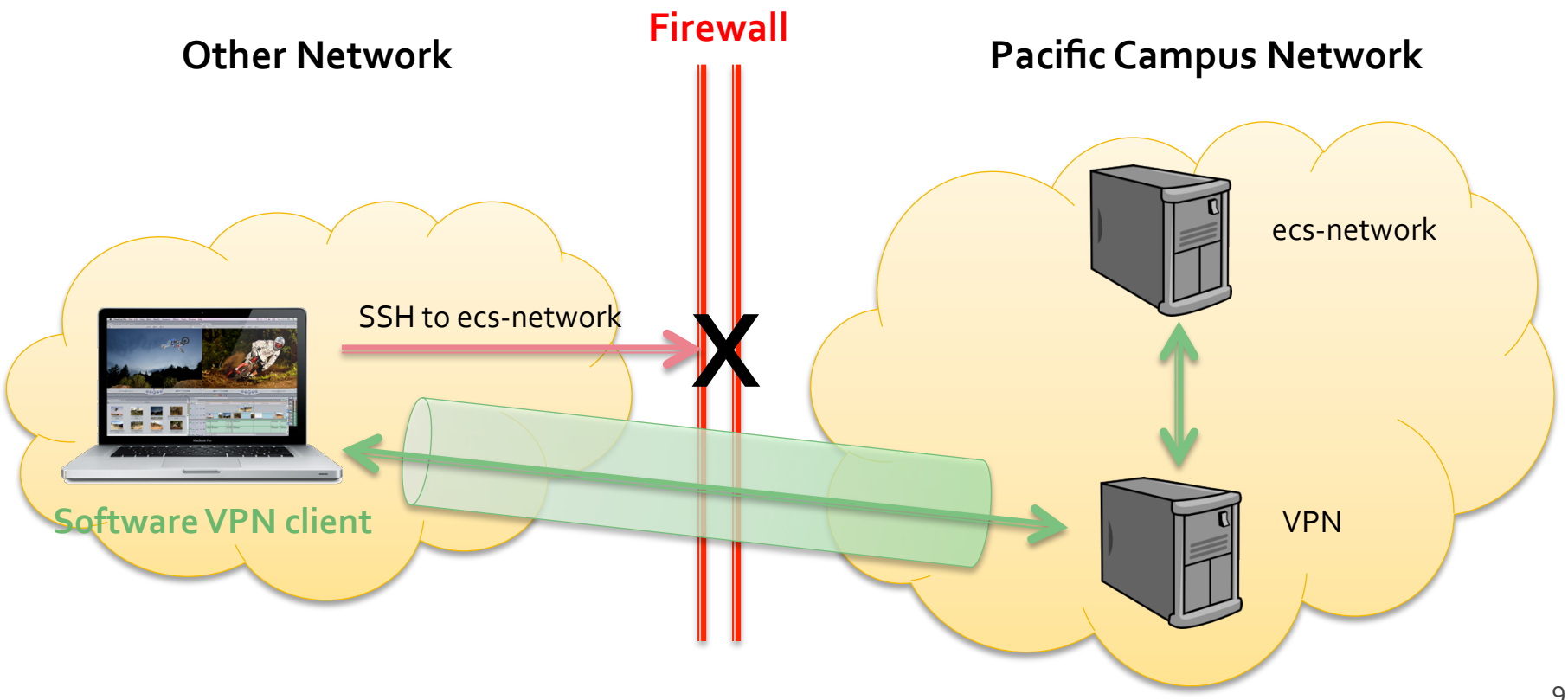
Off-Campus Access

- Two choices
- Choice 1:
  - *First*, SSH to `tiger.serv.pacific.edu` with your Pacific login (and X11 + compression)
    - "Tiger" is the only SSH server (AFAIK) that is accessible from off campus
  - *Second*, SSH from there to ecs-network

    ```
    ssh ecs-network.serv.pacific.edu -l <username> -C -Y
    ```
- Choice 2: Use a **VPN**

VPN

- Secure access to campus resources when off-network



Lab: Getting Started

1. Find a computer (yours or a lab)
2. Run the X-Windows server
    1. Xming on Windows, already running on Mac/Linux
3. Run the SSH program
    1. PuTTY on Windows, ssh on Mac/Linux
4. Connect to `ecs-network.serv.pacific.edu`
    1. Enable **SSH compression** (important for performance!)
    2. Enable **X-Forwarding** (otherwise, no graphics!)
    3. Username=whatever you selected on signup form
       Temporary password=`ecpe177#`
5. **Show me:**
    1. That you have **changed your password**: `passwd`
    2. That you have **Eclipse IDE running** : `eclipse &`

Fork command (i.e. run independently)

Lab: Creating a Project

- **Create an Eclipse C project** (not ANSI C)
  - New->C Project->Executable->Empty Project with GCC Linux toolchain
- **Use custom settings**: `-std=c99 -Wall -Wextra`
  - Setting options: Project -> Properties -> C/C++ Build -> Settings -> Tool Settings -> GCC C Compiler
    - **Warnings** tab: Check box for Wall
      - Turn on all warnings to *force* you to write better, safer C code)
    - **Miscellaneous** tab: Type in `-std=c99 -Wextra`
      - Use the more modern C99 standard
- **Add a folder** to put source code in: `src`

Lab: Simple Program

- Sample program:

```
#include <stdio.h>
int main() {
    printf("Tutorial demo program\n");

    for(int i=0; i<15; i++) {
        printf("Value of i: %i\n", i);
    }

    return (0);
}
```

- **Show me** that you can compile + run from inside Eclipse. Where is the program output?

Lab: Debugging

- **Show me** that you can Debug
  - Switch to the debugging mode (upper-right)
  - Add a breakpoint
  - Run to the breakpoint
  - Step into, step over, …
  - Switch out of the debugging mode (back to coding!)

Lab: Command Line Basics

- **Show me** that you can also compile + run your program at the command line

- Not familiar with command line basics? Review the Linux for programmers tutorial
  - http://bit.ly/cstutorials
  - Sections 1-3 are most relevant

Lab: Common Tasks ("strings")

- Use Eclipse to write a short program that:
  - Declare an array of 10 characters
  - After declaration, set the array to the string "testing"
    - **What needs to go after the "g" in "testing"?**
    - What function could help with this?
      Tip: http://en.wikipedia.org/wiki/String.h
  - Print the array
    - **What would printf do if the string in the array wasn't closed properly?**

Lab: Common Tasks ("strings")

```
#include <stdio.h>
#include <string.h>

int main(void)
{
  char myArray[10];

  strcpy(myArray,"testing");

  printf("%s\n", myArray);

  return 0;
}
```

Character Arrays

- Arrays of characters are ended with the NULL character (`'\0'`) - literally binary zero
- Most of the functions in `<string.h>` use the null character to identify end of string
  - strlen, strcpy, strcat, strtok, …
- printf() uses the null character when printing out a string
  - Ever wonder why you got garbage at the end?
  - `printf("My string: %s", myArray);`
- scanf() puts the null character at the end of strings
  - ```
    char word1[20], word2[20];
    scanf("%19s %19s", word1, word2);
    ```
    ```

Lab: Common Tasks (dynamic memory)

- Use Eclipse to write a short program that:
  - **Dynamically allocates** a two-dimensional array of integers (dimensions: 4 rows x8 cols)
    - Tip: see http://c-faq.com/aryptr/dynmuldimary.html
    - Check to see if memory allocation succeeded!
  - Fill the array with the numbers 1-25 (in whatever order you want)
  - Print the array
  - Frees **all** of the memory that was created
  - Exit
- Use **four functions**: createArray, fillArray, printArray, freeArray

Lab: Common Tasks (dynamic memory)

```
#include <stdio.h>   // Allows printf, ...
#include <string.h>
#include <stdlib.h>  // Allows malloc, ...
#include <errno.h>   // Allows errno

int** createArray(int rows, int cols);
void fillArray(int** myArray, int rows, int cols);
void printArray(int** myArray, int rows, int cols);
void deleteArray(int** myArray, int rows, int cols);

int main(void) {
    const int ROWS = 4;
    const int COLS = 8;
    int** myArray;

    myArray = createArray(ROWS, COLS);
    fillArray(myArray, ROWS, COLS);
    printArray(myArray, ROWS, COLS);
    deleteArray(myArray, ROWS, COLS);

    return EXIT_SUCCESS;
}
```

Lab: Common Tasks (dynamic memory)

```
int** createArray(int rows, int cols) {
    int **myArray;

    // Allocate a 1xROWS array to hold pointers to more arrays
    myArray = calloc(rows, sizeof(int *));
    if (myArray == NULL) {
        printf("FATAL ERROR: out of memory: %s\n", strerror(errno));
        exit(EXIT_FAILURE);
    }

    // Allocate each row in that column
    for (int i = 0; i < rows; i++) {
        myArray[i] = calloc(cols, sizeof(int));
        if (myArray[i] == NULL) {
            printf("FATAL ERROR: out of memory: %s\n", strerror(errno));
            exit(EXIT_FAILURE);
        }
    }

    return myArray;
}
```

Lab: Common Tasks (dynamic memory)

```
void fillArray(int** myArray, int rows, int cols) {
     int count = 1;

     for (int i = 0; i < rows; i++) {
          for (int j = 0; j < cols; j++) {
               myArray[i][j] = count;
               count++;
          }
     }

     return;
}

void printArray(int** myArray, int rows, int cols) {
     for (int i = 0; i < rows; i++) {
          for (int j = 0; j < cols; j++) {
               printf("%i ", myArray[i][j]);
          }
          printf("\n");
     }
}

void deleteArray(int** myArray, int rows, int cols) {
     for (int i = 0; i < rows; i++) {
          free(myArray[i]);
     }
     free(myArray);

     return;
}
```

Homework 3

- **Discuss assignment**
  - What are we doing?
  - What are the requirements?
- **Where to get programming help?**
  - Class resources page: http://ecs-network.serv.pacific.edu/ecpe-177/resources
  - Friends
    - Honor code: Homework is done individually
      - Allowed: Planning solution strategies, helping each other to debug programs…
      - Not allowed: Copying code from classmates

Homework 3

- Your PC – What software do I need?
  - Mac / Linux users – You're ready to go!
  - Windows users
    - Option 1
      - Install PuTTY (for SSH) and Xming (for X-Windows)
        - http://sourceforge.net/projects/xming/files/
        - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
    - Option 2 – **Recommended by past students!**
      - Dual boot Ubuntu (or install inside of VirtualBox) - 10GB disk
      - http://www.ubuntu.com/
      - http://www.virtualbox.org/