Projects Introduction

A lot to go over today…

- History of Linux
- Projects Overview
- Project partners
- Programming environment
- Programming language
- Useful Tools

History of Linux

The Beginning: Unix

- First implemented in AT&T Bell Labs, 1969.
- AT&T had to make a choice between using third party OS or developing their own.
  - Chose to implement own OS.
- Born from ideas and work performed on MULTICS OS.
- As a result of work on Unix (first implemented in the assembly language), C was born.

Time Line of Feature Introduction

- B-compiler, UNIX v1 – 1971
  - cat, chdir, chmod, chgrp, ed, mkdir, mkfs, mv, rm...
- C-compiler, Pipes, UNIX v3 – 1973.
- UNIX v5, open-sourced – 1974.
- sh, System V v1, UNIX v7. - 1979
- UNIX v10 (last edition) - 1989
- Somewhere between 1979 and 1989...
  - NFS, TCP/IP, STREAMS...

Standardizing UNIX – IEEE and POSIX

- POSIX – *Portable Operating System Interface for Computing Environments*
- What does this mean?
  - You can count on any modern operating system to adhere to this standard.
  - As long as you develop your programs by using functions available in the POSIX standard, “unistd.h”, your program will be portable to POSIX-compliant systems.

What's Included in the Standard?

- 1003.1 – System calls, library routines
- 1003.2 – Shell, basic UNIX (command-line) utilities
- 1003.3 – Test methods to demonstrate conformance
- 1003.4 – Real-time interfaces

Linux – Humble Beginnings

- Shortly after the final version of UNIX was produced, Linus appeared and published the first version of Linux.
- No OS at the time supported the Intel 80386 32-bit processors – Linus wanted to use his PC with that processor.
- It supported only his hardware – AT hard disks, Intel 80386.
- Since he was working on MINIX, some of the design was based off of MINIX.
- Started by porting bash(1.08) and gcc(1.40).
- For more details, refer to wikipedia or the book: *Just for Fun*.

Linux Today

- Current kernel version 3.12.6 (as of last week)
- Supports pretty much any platform and device the average user will interact with. Released to users as *distributions*, of which there are more than a hundred.

Distributions

- Ubuntu, Fedora, Slackware, SUSE, Red Hat, Debian, Gentoo, Mint, CentOS – all of these are distributions.
- Differences between distributions:
  - Package manager: aptitude, yum, portage, etc.
    - Used to install programs, libraries, documentation.
  - Kernel version: most are behind a few cycles
  - Windowing Interface: Gnome, KDE, etc.
  - Target audience: power-user, newbie, enterprise, etc.
  - Community

Which Distribution (Distro) to Use?

- The best advice I can give here is to use what you feel most comfortable using.
- If you haven't installed Linux on your computer before, maybe this class is the best time to give it a try!
- Other reasoning to choose one distribution over another:
  - Local standard - Colleagues/coworkers all use same distribution.

Additional References

- http://www.lwn.net/
  - Linux news site. Covers distros, conferences, and recent kernel development. Includes many links to free books, documentation, and the like.
- http://www.kernel.org/
  - Here's where you can obtain the latest Linux kernel, if you want to get your hands dirty.

Why Use Linux?

- Linux is open source
  - We actually have access to the kernel code and can change it
  - Much of the Internet runs on UNIX/Linux!
    - Wonderful time to get some experience

Unix/Linux Share

- Desktop/laptop – Linux 1.73%
- Mobile Devices – Android 79.0%
- Servers – Unix-like/Linux 66.8%
- Supercomputers – Linux 96.4-98%

Source:
http://en.wikipedia.org/wiki/Usage_share_of_operating_systems#Servers

Projects

Roughly Three Projects

- Write your own shell
  - Interface to the operating system
- Compile and modify an operating system kernel
  - Will have team virtual machines
- Create a program to read raw FAT file system images

Projects: Partners

- Projects will be completed in pairs
- Choose a partner and send me an email (diesburg@cs.uni.edu) with the name of your partner by 1/22.
- You need only submit one project per group
- Post on the eLearning forum to find a partner with compatible/complimentary skills and schedule

Demos

- Halfway Demo
  - Make sure you are on track
  - Chance for me to give you pointed help
- Final Project Demo
  - Demonstrate your project to me for points
  - I might ask either team member to describe code and design decisions

Programming Project

- Start projects when they're assigned.
  - They're often trickier than they look. Especially that synchronization project...
- Ask questions early.
  - If you're asking questions, be it to yourself or to others, you're thinking about the project. This will make it easier to complete them correctly and on time.
- Write small programs to test your program or language features you don't understand.

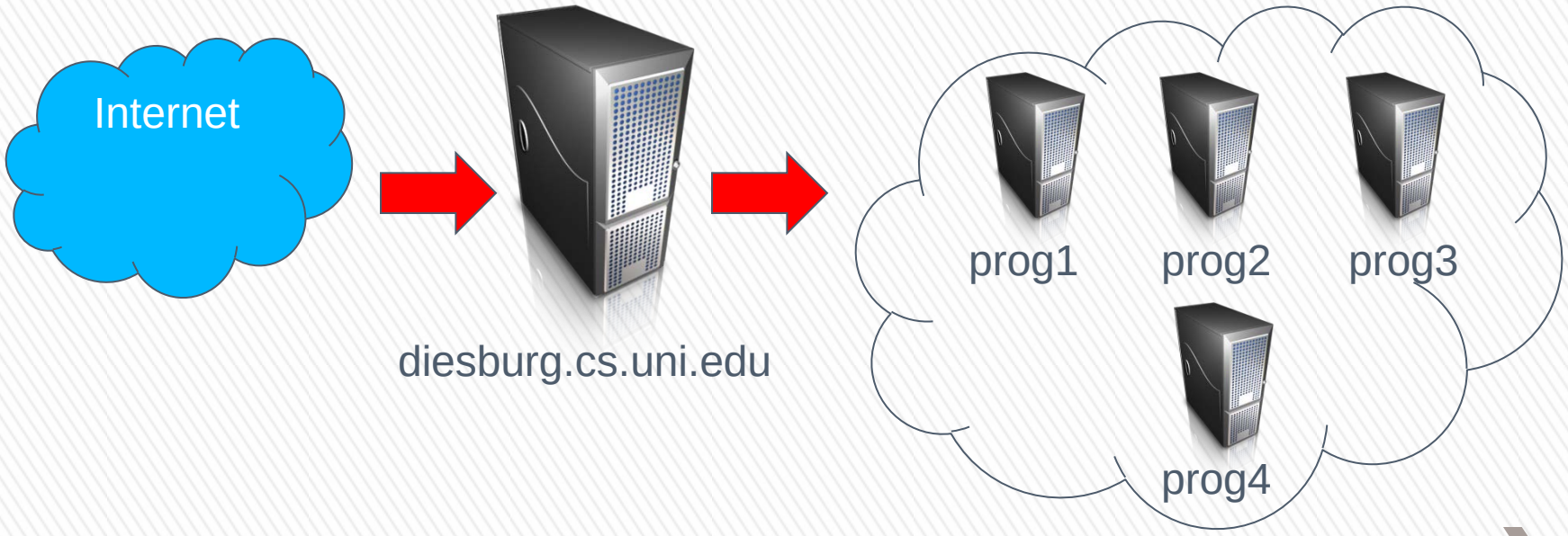
Programming Environment

- Project 1 and Project 3
  - Remote Linux servers
  - Accessible through ssh and server address diesburg.cs.uni.edu
- Project 2
  - Your own team virtual machines

Accessing the Remote Servers

- 4 Linux Servers at server address diesburg.cs.uni.edu
- Need usernames and passwords distributed in class
- If you are unfamiliar accessing remote Linux servers, please watch this video posted on today's webpage

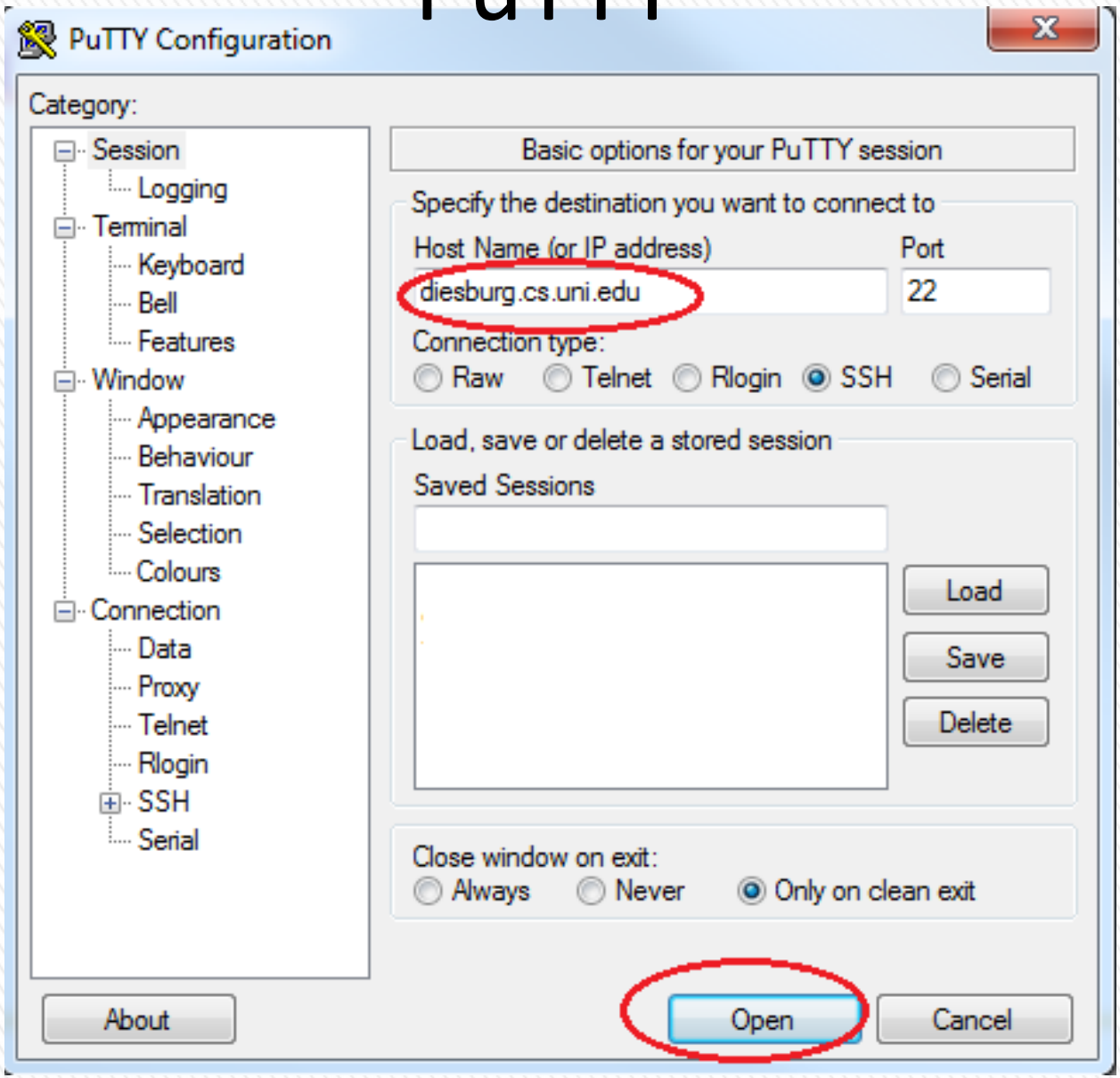
Server Visualization



Logging In

- Use SSH to connect to “diesburg.cs.uni.edu”
  - Secure SHell
  - If in Linux or OSX
    - Open up a command-line terminal
    - $> ssh <username>@diesburg.cs.uni.edu
  - If in Windows
    - You will need a terminal emulator
    - PuTTY (download from link on resrouces page)

PuTTY


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
diesburg.cs.uni.edu
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel

Once I am Logged In

- You will be logged onto the prog1 machine
  - But 3 other machines are at your disposal (prog2, prog3, prog4)
  - Might want to log into those machines if usage is too high
  - Can see the current system load and number of users by issuing the command ‘w’ at the prompt
- Going to another machine
  - At the prompt, use the ssh command:
  - $> ssh <username>@prog[2-4]
  - Example:
  - $> ssh diesburg@prog2
  - Use the same password that you used initially. Your files will be visible on all the machines

Next Steps

- Change your password to something you can remember
  - $> passwd
- Get familiar with Linux shell commands
  - Look at course “Resources” page under “Shell Resources”
  - Know at least the following
    - Maneuvering: cd, ls, pwd
    - Creating/deleting: touch, rm, rmdir, mkdir
    - Reading files: nano
    - Compilation: make, gcc
    - Packaging: zip, unzip
    - Help: man

Editing Source Files

- Two ways
  - Create and edit files on your own computer, then transfer to Linux server
  - Create and edit files directly on Linux server

- **I highly recommend the second way!**
  - File encodings from other operating systems can negatively effect compilations and cause very confusing errors
  - It’s not too bad, just pick a terminal editor

Editors -- Vim

- The vi editor was created by Bill Joy, the founder of Sun Microsystems when he was a graduate student
- The vim editor, vi improved, is the Linux version of the vi editor
  - multiple windows, highlighting text, and command history
- http://www.vim.org/

Editors -- Emacs

- GNU Emacs is an extensible, customizable text editor
  - Content-sensitive editing modes, including syntax coloring, for a variety of file types including plain text, source code, and HTML
- http://www.gnu.org/software/emacs/

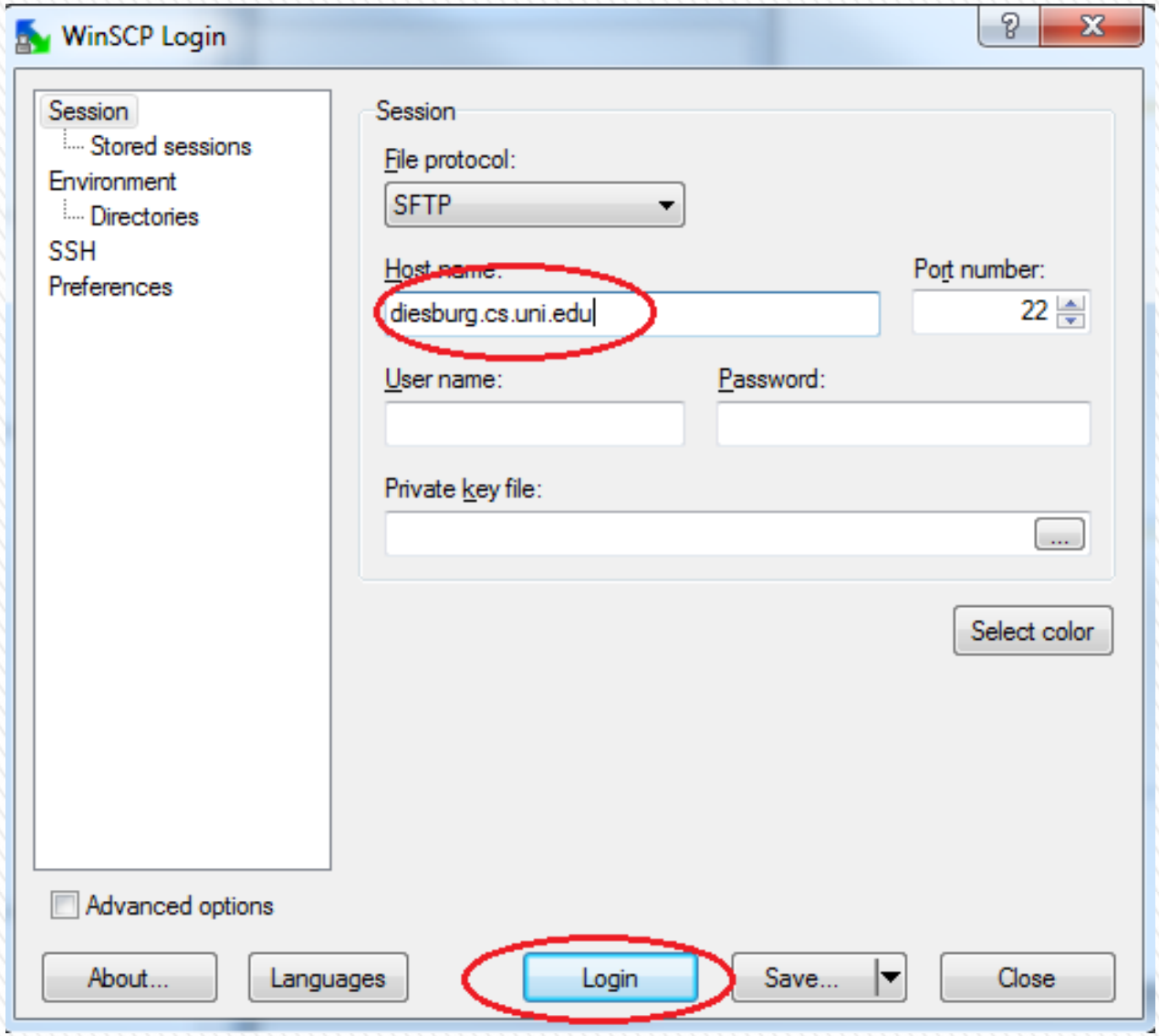
Editors -- Others

- Nano and/or pico are also available on most Linux systems
- If you have never worked in Linux before, this is your editor!
  - Extremely basic
  - $>nano <file name>

Transferring Files

- In Linux/OSX
  - scp
- In Windows
  - File transfer client like WinSCP
- From prog1
  - wget

WinSCP


WinSCP Login
Session
Stored sessions
Environment
Directories
SSH
Preferences
Session
File protocol:
SFTP
Host name:
diesburg.cs.uni.edu
Port number:
22
User name:
Password:
Private key file:
...
Select color
Advanced options
About...
Languages
Login
Save...
Close

Programming Language

- C is the programming language of operating systems
  - Kernel, system utilities, and large server programs (like apache and sendmail)
- Need to understand C to work inside the Linux kernel
  - Will get practice with C in project1
  - I will help, but you also need to get yourself up to speed with the basics

Quick C Language Tutorial

- Look in resources

Compiling

- Video
- $> gcc myfile.c –o myfile
  - gcc is the compiler
  - myfile.c contains my source code. It could be called anything as long as it ends with .c
  - -o is the output flag – the file that follows this flag will be the output executable
  - myfile – this is the output executable. Can be called anything

Running your executable

- $> ./myfile
  - ./ means “here” (will make more sense once we start the shell project
  - myfile is the name of the executable that you compiled

Part of Homework 1 (Due next Wed)

- Log onto the class servers
- Go through the online C tutorial
- To test your knowledge, create and compile a C program on the servers that
  1. Takes as input a string of up to 100 characters
  2. Counts every letter in the string
  3. Prints the letter occurrances

  (Hint – hash tables aren't part of the C standard library)

Useful Tools

manpages

- Extensive documentation that come with almost all Unix-like systems
- For documentation on C functions or packages
- Examples
  - $> man bash
  - $> man strncpy
- Sometimes multiple definitions, so use man section numbers
  - ‘man 1 printf’ shows bash printf
  - ‘man 3 printf’ shows C printf
- For more information on sections, see ‘man man’

zip

- Creating a zip file from folder proj1, which contains your source files:
  - $> zip –r proj1.zip proj1
- Unzipping a zip file
  - $> unzip proj1.zip

- Test this out before you submit a project!

Make

- **make**: A program for building and maintaining computer programs
  - developed at Bell Labs around 1978 by S. Feldman (now at Google)
- Instructions stored in a special format file called a "**makefile**".
- Will be provided for you for the first and second projects

Debuggers

- Debuggers let you examine the internal workings of your code while the program runs.
  - Debuggers allow you to set *breakpoints* to stop the program's execution at a particular point of interest and examine variables.
  - To work with a debugger, you first have to recompile the program with the proper debugging options.
  - Use the **`-g`** command line parameter to **cc, gcc,** or **g++**
    - Example: **`gcc -g -c foo.c`**

GDB, the GNU Debugger

- Text-based, invoked with:

  gdb [<programfile> <corefile>|<pid>]

- Issue ‘man gdb’ for more info

GDB Quick Start

```
$> ./my.x
$> Segmentation fault
$> gdb ./my.x
(gdb) run
… Segmentation fault
0x08048384 in main() at my.c:4
4                    *s = 'H';
(gdb) bt
#0 0x08048384 in main() at my.c:4
```