Class Projects and Environment

Networking

CS 3470, Section 1

Sarah Diesburg
A lot to go over today...

- History of Linux
- Programming environment
- Programming languages
- 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.
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 3.16.1 (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.
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:
Projects
Five Projects

• Modify a networking socket program
• Create network cables
• Create a full-featured chat program
  • This is the bigger project
• Advanced socket programming with shell scripting
• Analyzing packets
Programming Project

• Start projects when they're assigned.
  – They're often trickier than they look.

• 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.
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

Internet

diesburg.cs.uni.edu

prog1
prog2
prog3
prog4
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 resources page)
PuTTY
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
  • There is a video on the resources page to help
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
Programming Language

- C is the programming language of operating systems and the networking stack
  - Kernel, system utilities, and large server programs (like apache and sendmail)
- Need to understand some C for class examples
  - Will get practice with C in homework 1
  - 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” (or “this” version of the program)
  • 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
Useful Tools
manpages

• Extensive documentation that come with almost all Unix-like systems
• For documentation on C functions or packages
• Examples
  – $>\text{man bash}$
  – $>\text{man strncpy}$
• 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’
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!