19 – Final thoughts

CS 2043: Unix Tools and Scripting, Spring 2019 [1]

Matthew Milano
March 8, 2019

Cornell University
Table of Contents

1. Back to desktop environments

2. Linux everywhere

3. Linux everywhere
Back to desktop environments
Option 1: KDE

Figure 1: KDE
Option 2: GNOME

Figure 2: GNOME
Option 3: XFCE4

Figure 3: XFCE4
Tiling window managers

• Basic concept: windows don’t “float” or “resize”; they always take all available area
• Can sub-divide screen into smaller regions (in half, in quarters, etc).
• Applications automatically resize to snap to your new “grid”
• don’t move things with the mouse; change “splits” your screen with keyboard shortcuts
• some developers swear by these
• basically unused outside of serious developer circles.
Figure 4: xmonad
Figure 5: Awesome
Linux everywhere
Linux on windows: the old

- Git Bash uses MinGW / Cygwin
- Demo (I’m supposed to go to the cygwin website, and the git bash installer, then install git bash)
Linux on windows: the new

- A brief aside: Windows kernel is a federation of components called a *hybrid kernel*
  - Windows’ normal runtime is win32, built atop a core kernel layer
  - We won’t really get into what that means

- The Windows Subsystem for Linux is *magic*
  - new subsystem like win32
  - provides commands/API calls/syscalls from the Linux Kernel
  - *isn’t actually Linux*, but is close enough for all your userspace programs to not know the difference
  - You *definitely* want to install VcXsrv

- Demo: let’s look at WSL.
- There are lots and lots of guides to install online; google it!
Package management on windows: what for?

• You like automatic dependency management
• You like unified “app store style” auto-updating apps
• You want to install small, linux-like utilities or programs
• Three programs for three roles
  • Normal applications: Chocolatey
  • Libraries for C/C++/.NET software development: VCPkg or NuGet
  • Source-based package manager (from the folks at KDE): Craft
• Rapidly evolving space: there might be alternatives
  • google ’em!
• Demo time, I hope
Linux everywhere
What level of Linux do you want?

• Installing Linux on raw hardware best way to learn GNU/Linux
  • but easiest way to get stuck
  • simple things – like wifi, network, graphics – might not work at first
  • probably best not to do this in the middle of the semester

• Using your OS’s built-in Linux/Unix/Posix functionality often easier
  • A good alternative unless you’re doing really esoteric stuff
  • WSL on Windows / Terminal on Mac
  • If you’re on ChromeOS, then you already have Linux! Good choice.

• Integrating your “daily driver” OS with your shell is essential
  • Automatic on Linux — in fact hard to avoid
  • Not automatic on Windows, but not hard to do for modern apps
  • Mostly automatic on Mac, but you have to want it.

• VSCode demo on Windows
[1] Stephen McDowell, Bruno Abrahao, Hussam Abu-Libdeh, Nicolas Savva, David Slater, and others over the years. “Previous Cornell CS 2043 Course Slides”.