GNU/Linux & UNIX

Bryan Hoyle

October 3, 2016

UNIX History

- UNIX was created in 1969 as a alternative to a more complex operating system called Multics.
- Originally named UNICS (UNiplexed Information and Computing Service) as a play on Multics (Multiplexed Information and Computer Services), they later settled on UNIX.

Figure: Ken Thompson and Dennis Ritchie, two of the creators of UNIX

UNIX History (Cont.)

- It was created primarily by Ken Thompson, Dennis Ritchie, and Brian Kernighan
- 4 years later, UNIX was rewritten in C to make it easier to port to other machines.
- It was developed to be a more modular kernel architecture than previous operating systems.

GNU/Linux

- Later, a UNIX derivative called MINIX was created, but, frustrated with its licensing, Linus Torvalds creates a competitor called Linux.
- Originally, it used MINIX utilities but soon switched to GNU utilities creating a system that is called GNU/Linux

Figure: Linus Torvalds

Free Software vs Open Source

- Free Software (Free as in "Freedom" and free as in "free beer") means that the software has no restrictions for the user on source distribution
  - The GNU foundation created and propagates the GPL and the LGPL
- Open Source Software only guarantees that the source code is available
  - There are no guarantees for the user on distribution rights
- Both of these are positive things and each have their own unique set of pros and cons

UNIX Principles

- UNIX has a few core philosophies
- The most notable is "Do one thing and do it well."
  - Thus, UNIX-based systems use a collection of tools called "utilities," which are composable to be able to do more complicated tasks.

The Shell

- The most important utility is called the "shell."
  - This is the tool that manages input from the user and output from the programs they run
- The most common shells are derived from "sh," with the most popular being "bash."

Bash

- Bash is an abbreviation of Bourne Again Shell
- When started, it displays what is called a prompt

Prompt Example

```
bhoyle@Desktop /home/bhoyle $
```

- The first bit is the user and the machine name
- The next part is called the "current working directory"
- The dollar sign indicates where you can begin typing

SSH

- ssh is a tool that opens a shell on a remote machine
- This is incredibly useful for logging into zeus and mason and general remote work

Side note about paths

- If you are used to windows, you may be used to path names looking like `C:\Program Files\` or `D:\Documents and Settings\`
- In UNIX-like systems, however, the paths look like `/home/bhoyle` or `/mnt/usb/school`
  - Note that there are no drive letters nor are there backslashes
  - Separate drives are actually just directories in UNIX: there is no need for drive letters

Bash (cont.)

- To run a program in bash, one just types the name of the program into the shell

Command Example

```
bhoyle@Desktop /home/bhoyle $ pwd
/home/bhoyle
bhoyle@Desktop /home/bhoyle $
```

- `pwd` prints the current working directory

Basic Utilities

There are some basic utilities everybody should know:

`cd` foo	Changes the directory to foo
`cp` foo bar	Copies a file or directory from foo to bar
`ls` [foo]	Lists files in the current directory. If foo is given, it will list the files in foo.
`mv` foo bar	Renames a file or directory from foo to bar
`mkdir` foo	Creates a directory named foo
`rm` foo	Deletes the file foo
`rmdir` foo	Deletes the directory foo
`cat` foo1, foo2...	Reads the files foo1, foo2... to the screen
`man` foo	Gives a manual page on command foo

Live Demo

- Live demo time

Distributions

- A distribution is a managed way of distributing the Linux Kernel as well as a set of default software
- Different distributions generally have different goals and design principles
- Common distributions include Ubuntu, Linux Mint, Debian, Arch, and Gentoo
  - Personal Preference: Mint for beginners, Gentoo for more advanced users

Package Managers

- A package manager is a tool that manages installed applications and packages on a system
- Each distribution generally only has one package manager and everybody using that distribution uses the same one.
- Examples being that Ubuntu, Mint, and Debian all use apt, Arch uses pacman, and Gentoo uses portage.

Window Managers

- A window manager is the program which, surprise, manages your windows.
- On Microsoft Windows, this is not officially replaceable
- Responsible for the majority of how your system feels and looks
- Linux distributions come with a variety of window managers that fall into two general categories: tiling and non-tiling

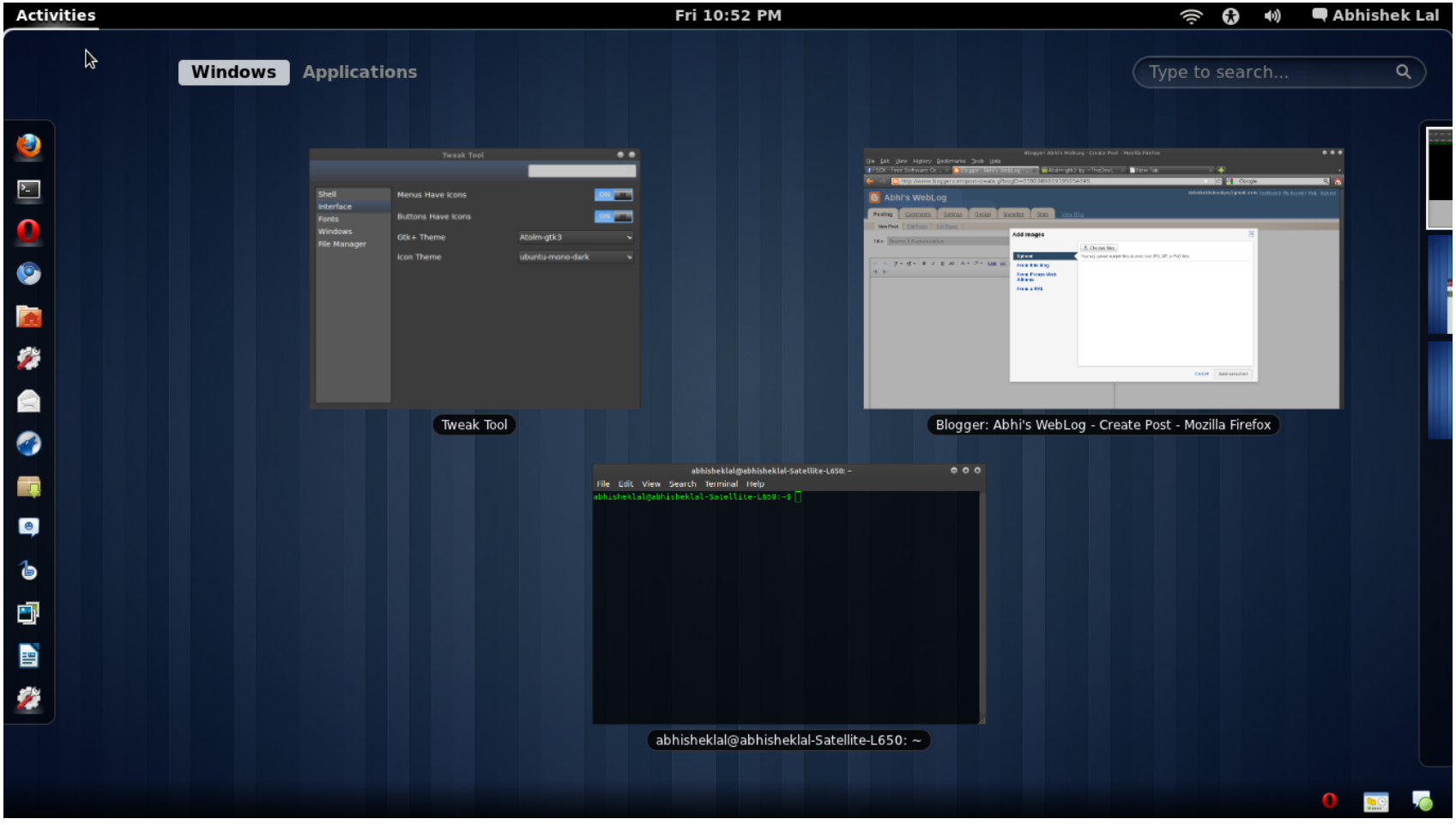
Desktop Environments

- Desktop environments supply things like the desktop icons and the menus
- This is a separate concept from the window manager which, as the name implies, only manages how the windows look and act

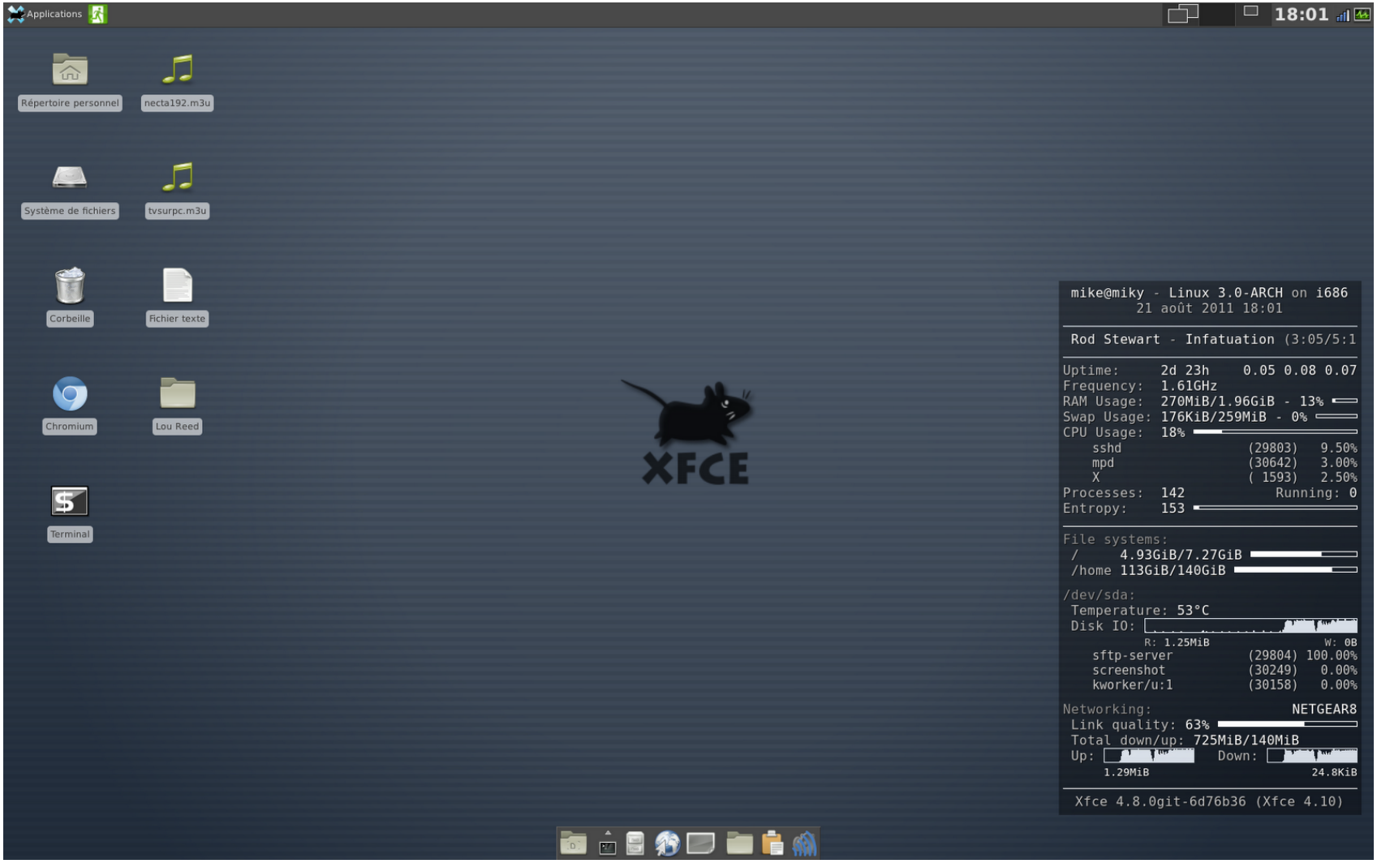
DEs with non-tiling WMs

- The most commonly used desktop environments with non-tiling window managers include Unity, GNOME 3, KDE, and XFCE
- Unity and GNOME 3 behave more like the Macintosh window manager with a command bar of some sort with virtual desktops and search commands.
- XFCE and KDE feel more like the MS Windows style with a start-menu like construct with primary focus on clicking icons

GNOME 3 Screenshot



XFCE Screenshot

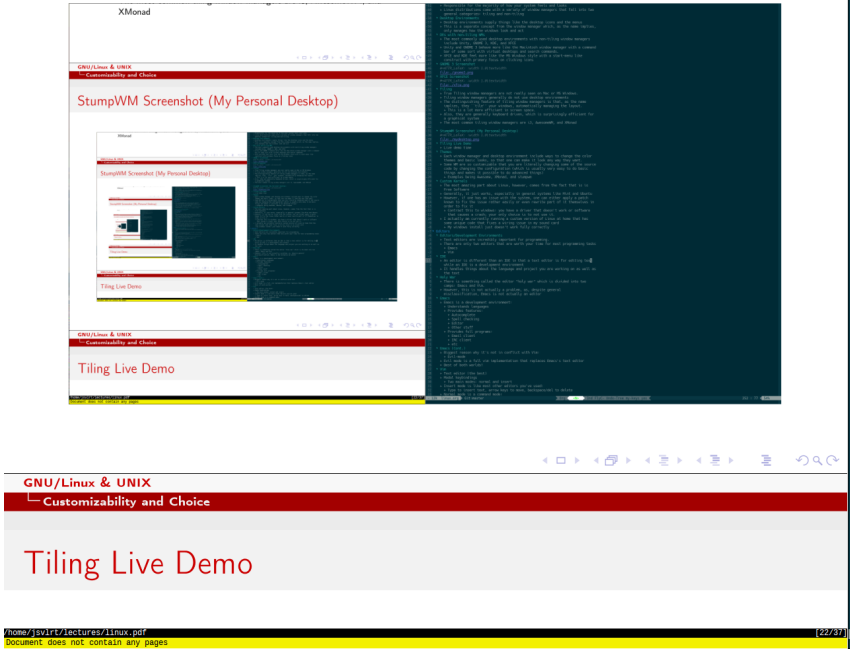


Tiling

- True Tiling window managers are not really seen on Mac or MS Windows.
- Tiling window managers generally do not use desktop environments
- The distinguishing feature of tiling window managers is that, as the name implies, they "tile" your windows, automatically managing the layout.
  - This is a lot more efficient in screen space.
- Also, they are generally keyboard driven, which is surprisingly efficient for a graphical system
- The most common tiling window managers are i3, AwesomeWM, and XMonad

StumpWM Screenshot (My Personal Desktop)

XMonad



Tiling Live Demo

- Live demo time

Themes

- Each window manager and desktop environment include ways to change the color themes and basic looks, so that one can make it look any way they want.
- Some WM are so customizable that you are literally changing some of the source code by changing the configuration (which is usually very easy to do basic things and makes it possible to do advanced things)
  - Examples being Awesome, XMonad, and stumpwm

Custom Kernels

- The most amazing part about Linux, however, comes from the fact that is is Free Software
- Generally, it just works, especially in general systems like Mint and Ubuntu
- However, if one has an issue with the system, one can either apply a patch known to fix the issue rather easily or even rewrite part of it themselves in order to fix it
  - Contrast this to windows: you have a driver that doesn't work or software that causes a crash; your only choice is to not use it.
- I actually am currently running a custom version of Linux at home that has some unique code that fixes a wiring issue in my sound card
  - My windows install just doesn't work fully correctly

Editors/Development Environments

- Text editors are incredibly important for programming.
- There are only two editors that are worth your time for most programming tasks
  - Emacs
  - Vim

IDE

- An editor is different than an IDE in that a text editor is for editing text while an IDE is a development environment
- It handles things about the language and project you are working on as well as the text

Holy War

- There is something called the editor "holy war" which is divided into two camps: Emacs and Vim.
- However, this is not actually a problem, as, despite general misclassification, Emacs is not actually an editor

Emacs

- Emacs is a development environment:
  - Understands languages
  - Provides features:
    - Autocomplete
    - Spell checking
    - Editor
    - Other stuff
  - Provides full programs:
    - Email client
    - IRC client
    - etc

Emacs (Cont.)

- Biggest reason why it's not in conflict with Vim:
  - Evil-mode
- Evil mode is a full vim implementation that replaces Emacs's text editor
- Best of both worlds!

Vim

- Text editor (the best)
- Modal keybindings
  - Two main modes: normal and insert
- Insert mode is like most other editors you've used:
  - Type to insert text, arrow keys to move, backspace/del to delete
- Normal mode is a command mode:
  - Each key does some command

Normal mode

- dd: Delete line
- yy: Copy (yank) line
- h,j,k,l: left,down,up,right
- i: insert mode
- a: go into insert mode after current character
- x: delete character
- much much more (beyond the scope of this lecture)

Insert mode

- ESC to go back to normal mode
- everything else types text

Live Demo

- Live demo time for Emacs/Vim

Other editors/IDEs

- jedit also works on Linux
  - Not very powerful
- IntelliJ Idea
  - Best IDE for java. This is honestly the only one you should use for real purposes, as it does most of the busy work for you.
  - Many classes have you use Dr. Java: this is because we want you to learn the language, not have an IDE do all the work for you.

Programming Language Support

- Almost every linux system has python built in now-and-days
- Every real linux system has the ability to easily install support for other languages:
  - gcc/g++: c/c++ compiler
  - ruby
  - scala
  - sbcl: common lisp compiler/interpreter
  - openjdk: java runtime/compiler
  - much more for haskell,php,racket,scheme,clojure,etc.,etc.
- Emacs has support for all of these languages and usually support for autocomplete and other language specific features

Live demo

- Compiling and running a c program
- Running a python program