Reflections on Teaching a Unix Class With FreeBSD

Lessons learned on both sides

Benedict Reuschling
benedict.reuschling@h-da.de

February 3rd, 2018

FOSDEM BSD Devroom

Table of contents

Lecture

In the beginning...

"What was it about UNIX that won my heart? ... UNIX is mysterious when you first approach. A little intimidating, too. But despite an unadorned and often plain presentation, the discerning suitor can tell there's lot going on under the surface."

– Love and UNIX: An Undying Affection by Thomas Scoville

My path to Unix

- In 2001, I started my CS education at the University of Applied Sciences, Darmstadt, Germany
- DOS/Windows background, some programming knowledge, knew very little about Unix
- My first programming lecture by Professor Schuette was an eye-opening experience: *"There are two groups in this department..."*

My path to Unix

- In 2001, I started my CS education at the University of Applied Sciences, Darmstadt, Germany
- DOS/Windows background, some programming knowledge, knew very little about Unix
- My first programming lecture by Professor Schuette was an eye-opening experience: *"There are two groups in this department. . . "*
  - He ran Linux on a Thinkpad, had all examples in terminal, little GUI exposure
  - Old school: made complicated things easy to understand
  - Challenging, yet rewarding, great mentor
  - took his Operating Systems course and many others after that, including Unix for Developers
  - Years later, he would tell me: *"Remember back when you started I used Linux and everyone else did, too? It's the same now that I'm on a Mac."*
- I started getting interested in Unix at insane levels, used Debian GNU/Linux for a few years
- Discovered the BSDs on my own one day, which lead me to where I am now
- I was asked to take over Unix for Developers, felt honored to be passed the baton

What is Unix for Developers?

- Undergraduate (Bachelor) elective CS course
- Yearly taught at the University of Applied Sciences, Darmstadt, Germany
- **Goals:** Teaching students how to do programming in/for a Unix environment, solving problems
- Required knowledge: Basic programming and Operating Systems
- Topics include:
  - Unix overview
  - Editors
  - Shell
  - Shell Scripting
  - Filesystems (focus on OpenZFS)
  - grep, sed, awk
  - Ansible
- 1.5 h lecture each week, 3 h labs every two weeks for students
- 5-6 lab sessions for hands-on experience, must be completed for final exam
- Written exam at the end of the semester to determine course grade

Changes I made to the course

- When I took over the course, slides were based on Linux 2.6
  - Slides needed updating anyway
  - Added more FreeBSD and ZFS content
- Originally held in German, now one of the few English language courses (3rd time now)
  - A lot of incoming exchange students must have courses in English
  - Most of the content and terms are in English anyway, no need to translate
  - Good training for students (and me) to flex the foreign language muscles
  - German students can answer in German or English, exchange students must use English
- Lab content was changed and extended
- It's never perfect: keep iterating and updating, use what works, remove what doesn't

Why bother?

- Unix skills are still relevant today
- We need to train the next generation early if we want to keep Unix alive
- It's an introduction, not indoctrination course
- Learn how to use the Unix tools and apply them in many other fields

Why BSD as a teaching tool?

- Opens the mind of students to other alternatives in the Unix space
- Compelling features like OpenZFS, boot environments, FreeBSD base system utilities
- Sources available under a license that allows exoloring and tinkering

Focus on Unix as a set of tools that can be applied to various problems: File processing using `awk`, shell scripting to create your own little utilites, etc.

How can you teach if you're not a professor?

- Universities are looking for practitioners that can cover certain topics they can't do
- You don't necessarily have to have a PhD or academic career to teach at universities
- Do what you love, what you are passionate about, and share it if you can
- In my case: combining my sysadmin work with the lecture:
  - Whenever I find something interesting or solve something all over again, I put it in a slide
  - Creates a reference for me if I need to remember how I solved something
  - Practical examples makes the lecture relevant and tangible
  - real world problems are demonstrated and solved

Challenges at the Start of each new Semester I teach the Course

- The inevitable question that will be asked every time:
  "Which Linux distribution are we going to use in this class?"
  - All of them and none!
- Two groups of students:
  - The ones who already know and use Unix/Linux
  - Those who don't or had only little prior exposure
- Don't bore the first group with topics they already know while not scaring off the second
- Ideally, there should be something new in my course for everyone (OpenZFS, Ansible, etc.)
- Lab 1 is there to bring everyone on the same page: installing FreeBSD from the shell
- Use your distro of choice if you can solve my labs on it (and you know what you're doing)
- If you never installed a Unix system before, here's what I use... (lead by example)

Labs

The Lab Contents

Lab contents vary depending on number of lectures per semester, new developments in the system, my creativity, and the fact that solutions are passed around between students.

Lab 1: Install FreeBSD in VirtualBox VM from the shell based on instructions provided. Some thinking required by students (device names, partition sizes, etc.)

Lab 2: Setting up an X environment (installing packages, editing configuration files) and basic desktop environment

Lab 3: Basic shell commands (what they do, chain of pipes, backticks, etc), introduction to creating graphs on the commandline (GraphViz)

Lab 4: Shell scripting (writing a `cdialog`-based menu for `beadm`), asking the user for a dataset and snapshot name and then create one using a script

Lab 5: Integrating grep, sed, awk into shell scripts (i.e. $1 in shell is different from it's use in `awk`), plotting data on the command line (filtering and preparing the data) using Gnuplot

Instead of having a dedicated lab for OpenZFS, a little bit is contained in each Lab (Snapshots, zfs compression comparison, etc).

Exam

Some notes on the exam (without giving away too much)

- Exam questions are a mix of practical problem solving, writing small programs, and knowledge repetition
- No student ever failed the class . . .
  - Some exchange students tried hard to fail last year
  - Writing answers with pencil and erasing everything (badly) before handing it in
  - Still legible enough, so graded as normal ;-)
  - They got enough points to barely pass (I wish them well!)

Fill in the blanks

Do you know the answer?

Doug McIlroy formulated three ideas that form the basic philosophy of Unix development. Complete the sentences below according to his principles:

- Write programs that do _____ and do _____.
- Write programs to _____.
- Write programs to handle _____, because that is a _____.

Fill in the blanks

Do you know the answer?
Doug McIlroy formulated three ideas that form the basic philosophy of Unix development. Complete the sentences below according to his principles:

- Write programs that do _____ and do _____.
- Write programs to _____.
- Write programs to handle _____, because that is a _____.

Write programs that do **one thing** and do **it well**.

Write programs to **work together**.

Write programs to handle **text streams**, because that is a **universal interface**.

Fill in the blanks

Do you know the answer?
Doug McIlroy formulated three ideas that form the basic philosophy of Unix development. Complete the sentences below according to his principles:

- Write programs that do _____ and do _____.
- Write programs to _____.
- Write programs to handle _____, because that is a _____.

Write programs that do **one thing** and do **it well**.

Write programs to **work together**.

Write programs to handle **text streams**, because that is a **universal interface**.

Here is what some students answered. It is not intended to publicly shame them (they'll remain anonymous anyway), but to show that grading can be fun when least expected.

Exam gold

(c) [3 Points] Doug McIllroy formulated three ideas that form the basic philosophy of Unix development. Complete the sentences below according to his principles:

- Write programs that do _____ and do _____.
- Write programs to sole problems.
- Write programs to handle bugs, because that is a problem.

(c) [3 Points] Doug McIllroy formulated three ideas that form the basic philosophy of Unix development. Complete the sentences below:

- Write programs that do stuff and do things.
- Write programs to Make ~~life~~ life easier.
- Write programs to handle everything, because that is a neccessity.

(c) [3 Points] Doug McIllroy formulated three ideas that form the basic philosophy of Unix development. Complete the sentences below according to his principles:

- Write programs that do ___good___ and do ___work___.
- Write programs to ___solve problems___.
- Write programs to handle ___memes___, because that is a ___holy duty___.

3 Points Doug McIllroy formulated three ideas that form the basic philosophy of Unix development. Complete the sentences below according to his principles:

- Write programs that do enough and do not to much.
- Write programs to automate processes.
- Write programs to handle this test, because that is a pain in the knee... yeah knee is about right

Things that I noticed and what students (indirectly) taught me

Things that I noticed

- Students grew up with GUI and touch interfaces, less and less with a terminal UI.
- Bash is perceived to be the default shell, becomes problematic when trying to teach shell scripting using `/bin/sh`

Things that students taught me

- You don't have to be perfect
  - Demos in lectures can fail spectacularly
  - Pronounciation can be wrong, students still understand what you mean
  - Loosen up a little, you don't have to be too strict
- Be surprised of what kind of solutions students can come up with

Future Topics and ToDo

- Lab: Sending desktop notifications from shell scripts for certain events (`libnotify`)
- Lecture: Slide-less presentation style
  - In theory, everything always works on slides, reality is different
  - Do demos during lecture to show how systems really do (or don't) work
  - Students can read the script containing more details afterwards
  - Invites discussion, direct comparison on student's own devices
- DTrace - Small introduction, warrants it's own lecture $\rightarrow$ TeachBSD[1]
- Performance: Measuring, Analyzing, Tuning (same as above)
- Kahoot-ing[2] the slides so that students can check their learning progress at their own pace

Suggestions welcome!

[1] `http://teachbsd.org`
[2] `https://kahoot.com`

Thanks for listening!

Questions, Comments?

Thanks for listening!

Questions, Comments?

Lecture material is available here:

https://www.fbi.h-da.de/organisation/personen/reuschling-benedict/unix-for-software-developers.html