My experience in using *Linux From Scratch* as an advanced exercise

Randolph Langley
Florida State University
langley@cs.fsu.edu

October 17, 2016

Abstract

A summary of my experience and lessons learned from using Linux From Scratch installation as an advanced exercise over the last two semesters.

1 Introduction

It was clear early on that system administrators had to be rabid jacks of all trades. – page 1269 ULSAH [2]

In summer 2016, I came up with the idea of having students do a complete installation of Linux From Scratch as an optional class project. In a class of 30-odd students, 6 students completed this optional project.

I found this quite encouraging. I think that doing an installation of Linux From Scratch (LFS) is an outstanding way to learn exactly how a Linux system is put together. Rather than just learning about a single aspect of system configuration, I find the depth and breadth of this project to be a completely different experience for the students.

LFS illustrates a complicated process for building a shared library system, not only giving great detail about the internals but also the philosophy behind building shared libraries.

2 Outline of talk

2.1 Philosophy

System administration can often seem like it involves much configuration of complex software, and from that laborious experience we hope to abstract out the internal structure of such complex systems.

I like to try to motivate the study of system administration by expounding on the model of a kernel which provides system calls feeding libraries, which provide APIs for userland processes. I find that this more strongly motivates real-world troubleshooting techniques based on strace and ltrace.

Using LFS as a large scale exercise illuminates the orthogonal problem of the complex process of building any new dynamically linked library system, not only giving in great detail the internals of cur-

rent GNU/Linux practice, but also the philosophy of building a new clean-room toolset and environment.

2.2 A summary of the LFS build exercise

There are too many other good reasons to build your own LFS system to list them all here. In the end, education is by far the most powerful of reasons. As you continue in your LFS experience, you will discover the power that information and knowledge truly bring.
– Gerard Beekmans, LFS 7.10 Foreword

LFS, unlike a managed package distribution like Red Hat or Debian, or even pure source distributions like Sabotage, is based on building a system from original source by following a book, *Linux From Scratch [1]* (currently version 7.10). The book is composed of nine chapters:

- Chapter 1 Introduction
- Chapter 2 Preparing the Host System
- Chapter 3 Packages and Patches
- Chapter 4 Final Preparations
- Chapter 5 Constructing a Temporary System
- Chapter 6 Installing the Basic System Software
- Chapter 7 System Configuration
- Chapter 8 Making the LFS System Bootable
- Chapter 9 The End

The LFS system is based on 72 packages from various organizations; 36 of them are from gnu.org websites, and 7 are from kernel.org. Three packages are hosted at linuxfromscratch.org itself. There are also 8 patch files.

Building the system starts with creating one or more partitions to host the build (Chapter 4), and then compiling and installing the fundamental 12 toolchain steps in the new partition(s) in Chapter 5 (diagram 1.)

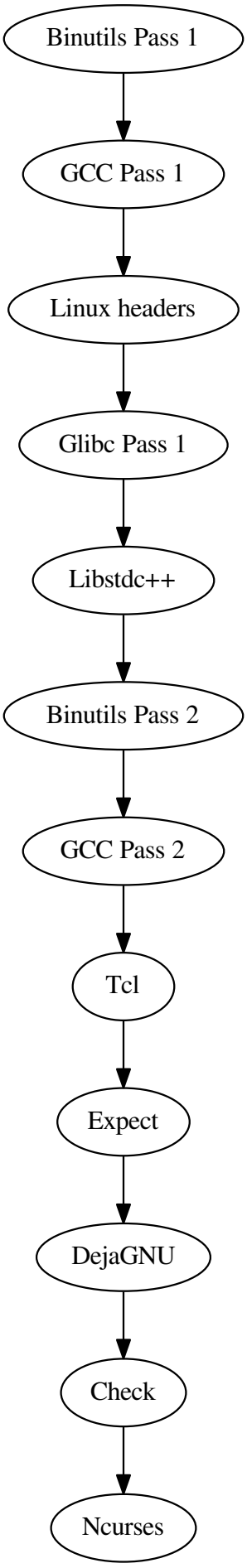

Diagram 1: Chapter 5's fundamental toolchain

After that, one compiles the remaining 19 base toolchain elements: Bash, Bzip2, Coreutils, Diffutils, File, Findutils, Gawk, Gettext, Grep, Gzip, M4, Make, Patch, Perl, Sed, Tar, Texinfo, Util-linux, Xz.

In Chapter 6, one first sets up and then enters the basic chroot environment. The chroot ensures that we are both only modifying the new build's directories, and that we are using initially only the new toolset from Chapter 5 and those bits that are being incrementally added as we work through Chapter 6.

Chapter 6 has 63 steps that work with source code, and 7 that work with the configuration of the new build.

After first installing the Linux headers and the man pages, the first compilation is Glibc, followed by the section "Adjusting the Toolchain" which makes sure that the remaining compilations are working against the just-compiled Glibc.

Next the file utility is compiled, and then Binutils is recompiled for the third time, followed by GCC's third compilation. Then we are off to the races: 59 packages are then compiled, with the eight downloaded patches applied amongst a vast motley of configuration options, tests, and path adjustments (it's not all just `./configure --prefix=/usr && make && make install`), ending with the compilation of Vim.

In Chapter 7 final configuration is done to the system; Chapter 8 finishes the exercise with creating `/etc/fstab`, compiling a new Linux kernel, and running GRUB. Chapter 9 congratulates the installer and invites the installer to join the 26,000+ people who have listed themselves as having completed the exercise.

2.3 How I structure this

I have the students do this exercise with QEMU virtualization, with the host system as Salix (an outstanding Slackware distribution that I only recently found.) The students install Salix to one image file, and do the LFS installation on a separate image file. The final step is booting the new LFS installation alone.

2.4 The value of this exercise

Obviously, anyone doing this exercise will learn a tremendous amount about every level of the operation of Linux: this exercise covers literally everything from building a kernel to configuring every major `/etc` file and directory on a GNU/Linux system.

Beyond that obvious value, doing the LFS exercise in the timeframe of an exercise in a single class is a strenuous endeavor. Attention to detail is key, as is developing an understanding of the nuances of the structure of the complex dynamically-linked GNU/Linux monoculture. While much of the keyboard work can be done as an exercise in cut-and-paste from the LFS book to Bash running in the chroot environment, that's not uniformly true, and learning to recognize those more challenging areas is one of the better lessons available in this exercise.

LFS illustrates a complicated process for building a new dynamic library system, not only giving in complete detail the internals but also the philosophy of building such systems. It also discusses the philosophy of package management from the most fundamental angle: here we have a lot of source code which we will compile and install, but how do we *maintain* the forthcoming flood of changes that

will develop in the future? Often these changes are engendered by a CVE and in many cases should be dealt with immediately.

It also gives one perspective on the ongoing debate about the trade-offs involved in dynamic linking versus static linking, and the value of alternative C libraries, such as MUSL.

3 Conclusion and next steps

While this exercise will stimulate a lot of interest from some students, many of whom will learn a great deal from doing this, inevitably some students will be overwhelmed by such a challenge. I have tried to motivate students to start working on this early by giving more credit for finishing early, since anyone who procrastinates on this exercise is very unlikely to finish it in a blaze of last-minute frenetic "configure and make."

The next step obviously might be an exercise in *Beyond Linux From Scratch*, which would give deeper insight into the complexities of source code interactions and localization issues. However, I think an exercise in applying package management *a la* the issues introduced in 6.3 would be even more interesting since it involves many philosophical trade-offs – just look at how many different package management systems exist in the GNU/Linux world: from RPM and Debs to Sabotage Linux's `butch`, the panoply is worth exploring.

Another angle that I personally find interesting is looking at systems like Alpine Linux, sta.li, and Sabotage Linux, which are innovating the Linux landscape with a move to MUSL and a strong interest in static linking.

References

[1] Gerard Beekmans, edited by Bruce Dubbs, *Linux From Scratch, version 7.10*, `http://www.linuxfromscratch.org/lfs/view/stable/`, 2016.

[2] Evi Nemeth, Garth Snyder, Trent Hein, Ben Whaley, *et al*, *Unix and Linux System Administration Handbook*, 4th Edition, Prentice Hall, 2011.

[3] *Alpine Linux*, `https://alpinelinux.org/`.

[4] *STALI: Static Linux*, `http://sta.li/`.

[5] *musl libc*, `https://www.musl-libc.org/`.