Making Your Own Tiny Linux

by
Aaron Grothe
OLUG - June 2015

What do you really need?

Systemd? - Nope
Thousands of utilities - Nope
Linux Kernel - Yep
Basic Filesystem - Yep
Some support files - Yep

Some of the Tools you Can Use

BuildRoot
OpenWRT BuildRoot
Linux From Scratch
Yocto Project
GeexBox

Tools have the benefit of providing an easily repeatable process, the ability to do automated builds/updates, etc. You are also gaining a lot of stuff you may not need/want

BuildRoot

BuildRoot - http://www.buildroot.org

Takes care of some of the pieces - Cross-compilation environment / downloading sources etc

Updated every 3 months

Has pre-set up templates for things like Raspberry PI/Beaglebone, etc

Used by some people I know

OpenWRT BuildRoot

Heavily modified version of BuildRoot

Used by the OpenWRT people to compile their software for all architectures/target routers

Is impressive in scale

Linux From Scratch

Linux From Scratch - http://www.linuxfromscratch.org

Very nice book on how to do your own Linux system from source

Follow through with it and you'll have a complete system by the end of it

ALFS - Automated Linux From Scratch / BLFS / Hardened, etc

Also has nice book Cross Linux From Scratch

Yocto Project

Yocto Project - http://www.yoctoproject.org

A Linux Foundation project

Designed to help create custom Linux images for embedded systems

GeexBox

GeexBox - http://www.geexbox.org

GeexBox is a system for generating a LiveCD that plays media stored on the CD. Can also do a lot of other things as well

Has a very interesting little build system. When doing this years ago for a company I worked for this was about the only one that would result in a workable image without huge investments of time/effort

How to Build a Minimal System

Years ago Bruce Perens put together a very nice little 3-part article that covered the basics of creating a tiny Linux system

Here are the links to the 3 parts

http://www.linuxjournal.com/article/4335 - Part 1
http://www.linuxjournal.com/article/4395 - Part 2
http://www.linuxjournal.com/article/4528 - Part 3

Minimal System

Designed to create a bootable floppy and talked a bit about how some of the initial Debian boot floppies worked

Linux Kernel
BusyBox
SysLinux
InitRd - Initial Ram Disk - compressed file system

Linux Kernel

How small can you get a Linux 4.0 kernel nowadays?

If you do make defconfig and set it to do everything static you will probably end up with a 5mb bzimage kernel

Of course that includes a lot of drivers/etc you don't need/want

Still hard to get it down to the size where you can get it on a floppy nowadays with a minimal file system

BusyBox

BusyBox is a very cool project that puts a load of utilities/software into one executable and depending on the name you use to call it does that work.

E.g. Busybox - has a version of vi, ls, which, who, and so on all in one executables

Many routers/embedded systems use busybox for the majority of their functionality

BusyBox is also one of the primary GPL defenders

SysLinux

SysLinux is a simple boot loader that can load linux from a fat/fat32 filesystem. Used for a lot of mini-distributions

InitRD

Initial Ram Disk

Compressed filesystem that is presented to the bootloader along with the Linux Kernel

Generated by the Genromfs command. Can also be built by hand.

Contents - /dev /bin/ /sbin /etc /home /var that will be available in the system

My Build System

Created a basic Debian 8.0 (Jessie) i386 system via kvm

Didn't want to deal with cross-compiling
Wanted to be able to drop the whole system onto a big usb drive and take it with me
Wanted to be able to pull utilities from Debian during testing

What am I Building?

Years ago 15+ now there was a project called Tinfoil Hat Linux
Tinfoil Hat Linux is/was a bootable floppy
It was designed for you to be able to use it to gpg encrypt/decrypt data
Hasn't been updated in years, schmoo.org site has been down for some time

What am I Building?

TFL has several nice security features

- Background encrypt/decrypt throwing things off
- Paranoid mode (grey screen)
- GPGgrid matrix based input of password
- Flipping LEDS off/on keyboard in morse to make listening electronically more interesting
- Checks the bios signature of machine to make sure it hasn't been altered
- No network support compiled into the Linux Kernel

What I'm Building

Tin Fez Linux - is a spiritual successor to TinFoil Hat Linux

Will use updated Linux Kernel (ext 3 & 4)/BusyBox/GPG
Use mdev from Busbox for ability to dynamically detect hardware
Some of the components are still being researched morse2led doesn't work on newer kernels so looking into using setleds, etc.
Will be 10mb in size possibly
Designed to be put on a write-protectable USB drive. That way you can control when stuff is able to be written to the image

What I'm Building

When is it going to be done?

Good question. I'm still doing some work on the base system and getting pieces pulled over from the original system so it probably won't be for a while yet.

Will probably put it up at Sourceforge as a couple of subprojects

Tips

Get your kernel right - Need initrd support in the kernel, need ramfs support and several other options to work
Make sure to set your partition to bootable. I spent several hours before I figured that out
Start with the defaults for most things and start customizing after that

Tips (Cont)

If you do a KVM setup like mine you can mount the target image in the build system, update and boot it without making changes

e.g. kvm -hda tflbuild.img -hdb boot.img # boots system mount/write any data to the boot.img filesystem; do a sync then start up the kvm with the boot.img while the original is still running

kvm -hda boot.img

saves a lot of time

Demo of Tinfoil Hat Linux

Quick demo to show off some of its components

paranoid options/menu, etc

Demo of My Tiny Linux

Just a quick demo to show what it looks like and how it works a bit

This version doesn't have the Tin Fez components on it as I'm still working on those

Demo of the Build Environment

Quick look at the build environment

Just to show how parts of it are laid out. Still have quite a few shell scripts to write to automate parts of it

Summary

You learn a lot building your own Linux distribution
The core components haven't changed that much in the last 15 years
Tools like BuildRoot and Yocto make things a lot easier
Some of the old guides/info out there from projects like Eagle Linux can also be helpful as well
VMs have made the process of testing the resulting distros a lot more pleasant