Current release 0.8.11 (April 8, 2024)

What is toybox?

Toybox combines many common Linux command line utilities together into a single BSD-licensed executable. It's simple, small, fast, and reasonably standards-compliant (POSIX-2008 and LSB 4.1).

Toybox's main goal is to make Android self-hosting by improving Android's command line utilities so it can build an installable Android Open Source Project image entirely from source under a stock Android system. After a talk at the 2013 Embedded Linux Conference explaining this plan (outline, video), Google merged toybox into AOSP and began shipping toybox in Android Marshmallow in 2015.

Toybox aims to provide one quarter of a theoretical "minimal native development environment", which is the simplest Linux system capable of rebuilding itself from source code and then building Linux From Scratch and the Android Open Source Project under the result. In theory, this should only require four packages: 1) a set of posix-ish command line utilities, 2) a compiler[1], 3) a C library, and 4) a kernel. This provides a reproducible and auditable base system, which with the addition of a few conviencences (vi, top, shell command line history...) can provide a usable interactive experience rather than just a headless build server.

Why is toybox?

The 2013 toybox talk at ELC was devoted to this question, and has the following sections:

- 0m29s The smartphone is replacing the PC
  - 4m22s Software needed to become self-hosting
  - 6m20s Do we care if android or iphone wins?
- 9m45s Android not vanilla: oppose or accept?
  - 11m30s Open source can't do User Interfaces
- 15m09s Android is not copyleft: oppose or accept?
- 18m23s Security issues
- 21m15s Solutions to the software problems
- 22m55s What toybox needs to be/do
- 28m17s What is toybox?
  - 28m58s Why toybox started...
  - 37m50s What does toybox actually implement?

A more recent talk from 2019 compares BusyBox vs toybox and explains the design decisions behind both. (A 2015 toybox talk was part of the channel accidentally deleted off youtube by the Linux Foundation, but the outline is still available.)

What context was toybox created in?

The toybox maintainer's previous minimal self-hosting system project, Aboriginal Linux, got a native development environment down to only seven packages in its 1.0 release (busybox, uClibc, gcc, binutils, make, bash, and linux) and then built Linux From Scratch under the result. That project was the reason toybox's maintainer became busybox maintainer, having done so much work to extend busybox to replace all the gnu tools in a Linux From Scratch build that the previous maintainer handed over the project (to spend more time on buildroot).

Despite the maintainer's history with busybox, toybox is a fresh from-scratch implementation under an android-compatible license. Busybox predates Android, but has never shipped with Android due to the license. As long as we're starting over anyway, we can do a better job.

Toybox's current minimal native development environment builder is a new tiny implementation integrated into the toybox source. The "make root" target will create a simple toybox chroot (by default in the root/host directory), and adding a LINUX= argument to the make command line pointing to Linux kernel source code creates a tiny bootable system with a wrapper script to run it under the emulator qemu.

The list of commands remaining before we can build Linux From Scratch under the result (with an appropriate compiler) is tracked in the roadmap, and doing so is one of the main goals for toybox's 1.0 release.

Building LFS requres fewer commands than building AOSP, which has a lot more build prerequisites. In theory some of those can be built from source as external packages (we're clearly not including our own java implementation), but some early prerequisites may need to be added to bootstrap AOSP far enough to build them (such as a read-only version of "git": how does repo download the AOSP source otherwise?) [2]

What commands are planned/implemented in toybox?

The current list of commands implemented by toybox is on the status page, which is updated each release. There is also a roadmap listing all planned commands for the 1.0 release and the reasons for including them.

In general, configuring toybox with "make defconfig" enables all the commands compete enough to be useful. Configuring "allyesconfig" enables partially implemented commands as well, along with debugging features.

Relevant Standards

Most commands are implemented according to POSIX-2008 (I.E. The Single Unix Specification version 4) where applicable. This does not mean that toybox is implementing every SUSv4 utility: some such as SCCS and ed are obsolete, while others such as c99 are outside the scope of the project. Toybox also isn't implementing full internationalization support: it should be 8-bit clean and handle UTF-8, but otherwise we leave this to X11 and higher layers. And some things (like $CDPATH support in "cd") await a good explanation of why to bother with them. (POSIX provides an important frame of reference, but is not an infallable set of commandments to be blindly obeyed. We do try to document our deviations from it in the comment section at the start of each command under toys/posix.)

The other major sources of commands are the Linux man pages, the Linux Standard Base, and testing the behavior of existing command implementations (although not generally looking at their source code), including the commands in Android's toolbox. SUSv4 does not include many basic commands such as "mount", "init", and "mke2fs", which are kind of nice to have.

For more on this see the roadmap and design goals.

Download

This project is maintained as a git archive, and also offers source tarballs and static binaries of the release versions.

The maintainer's development log and the project's mailing list are also good ways to track what's going on with the project.

What's the toybox logo image?

It's carefully stacked soda cans. Specifically, it's a bunch of the original "Coke Zero" and "Pepsi One" cans, circa 2006, stacked to spell out the binary values of the ascii string "Toybox", with null terminator at the bottom. (The big picture's on it's side because the camera was held sideways to get a better shot.)

No, it's not photoshopped, I actually had these cans until a coworker who Totally Did Not Get It tm threw them out one day after I'd gone home, thinking they were recycling. (I still have two of each kind, but Pepsi One seems discontinued and Coke Zero switched its can color from black to grey, presumably in celebration. It was fun while it lasted...)

Footnotes

[1] Ok, most toolchains (gcc, llvm, pcc, libfirm...) are multiple packages, but the maintainer of toybox used to maintain a fork of tinycc (an integrated compiler/assembler/linker which once upon a time did build a bootable linux kernel before its original developer abandoned the project), and has vague plans of trying again someday. The compiler toolchain is _conceptually_ one package, implementable as a single multicall binary acting like make, cc, as, ld, cpp, strip, readelf, nm, objdump, and so on as necessary. It's just the existing packages that do this ~~kinda suck~~ don't. (In theory "make" belongs in qcc, in practice llvm hasn't got its own make so toybox probably needs to add it after 1.0 to eliminate another gpl build prerequite from AOSP.)

[2] The dividing line is "Is there an acceptably licensed version Android can ship, or do we have to write one?" Since android is not "GNU/Linux" in any way, we need to clean out all traces of gnu software from its build to get a clean self-hosting system.