stali – static linux

Anselm R Garbe
anselm@garbe.us

Abstract

stali is a new and unique linux distribution that consists of static executables only (with some exceptions).
stali proves that statically linked executables start faster, are smaller, use lesser memory and are easier to update.
Keywords: linking; static linking; linux; stali

1 Introduction

stali[1] is a very simple linux distribution, it consists of a minimalist init system, a radically cleansed file system structure, a hand selected collection of the best open source tools for each task, and does not need package management.
Its primary focus is the creation of small and fast static executables, including a small monolithic linux kernel that needs to be customised for each specific system.
stali uses its own unique build chain around the Plan 9 make tool *mk*[6]. Its build chain completely disregards the usual *configure*[4]-based approach to build open source packages, mainly because this approach is too slow and unflexible for building static executables and libraries. The build chain is the real value of stali, because it enables developers to also cross-compile for embedded devices.

2 Size

Contrary to the expectation, static executables can be a lot smaller than their dynamic counterparts, if one overcomes the use of bloated libraries at linkage time.
Linking a stripped hello world program with *glibc*[3] results in a 600kb executable. Linking it with *uClibc*[2] results in a 7kb executable.
Statically linking stali's default Korn shell[5] (ksh) with *uClibc* results in a 170kb executable; linking it with *glibc* dynamically results in a 234kb executable. This example scales for the whole core userland of stali. Nearly all static executables are smaller than their dynamic counterparts in common linux distributions.
Another aspect of static executables is that they are size-optimised at linkage time. Static executables do not contain complete static libraries but just those objects from a library archive that expose required symbols. Dynamically linked executables or libraries can't be size-optimised in a similar fashion.

3 Memory footprint

Due the observation that static executables are generally smaller, the overall memory footprint is less. stali does not have dynamic libraries that are loaded into memory – even if only tiny portions of a dynamic library are used.
Since the size of static executables outnumbers the theoretical overhead of statically linked library functionality, the aspect of cloned objects is neglectible.

4 Performance

The only measurable difference between static and dynamic executables is the start up time. Benchmarks published on the stali website[1] show that a performance gain of 400% and more is common to static executables. The slow start

of dynamic executables is related to the system's symbol resolution.

5 Security

Several people argue (with implicitly requiring ABI-stability) that dynamic executables benefit from security fixes in dynamic libraries they depend on. This is true, however the opposite is also true: if there is a security flaw in a dynamically linked library, all programs that depend on it are affected; whereas statically executables are not.

Another argument often heard is that static library functions have predictable addresses, whereas dynamic linking provides the ability of address randomization. There are two answers to this.

The first is: Technically it is possible to use platform-independent code in static executables and assuming the kernel supports address randomization for executables we have a similar feature.

The second is: In reality, address randomization is predictable and we usually see the same addresses when a dynamic library is loaded or has been preloaded reproducibly.

Apart from that stali tends to link against libraries with low footprint, such as *uClibc* instead of *glibc* when possible. This leads to a higher probability of lesser vulnerabilities in the dependent library, simply because lesser code contains fewer bugs.

6 Durability

Static executables are durable and will run on the same platform in a similar kernel environment for years. They aren't affected from ABI changes like dynamic executables.

7 Updates

Updating a static userland only requires to replace the static executables in the filesystem. There are no side-effects of already running processes like in a dynamic userland.

8 Conclusion

stali proves that a static linux distribution provides several advantages, in particular being smaller, consuming lesser memory, starting faster and being more secure.

Apart from this, stali's build chain is also interesting for embedded device development and cross-compiling.

References

[1] `http://sta.li` (stali website)

[2] `http://www.uclibc.org/`(uClibc website)

[3] `http://www.gnu.org/software/libc/`(glibc website)

[4] `http://www.gnu.org/software/autoconf/`(autoconf website)

[5] `http://en.wikipedia.org/wiki/Korn_shell`(Korn shell wikipedia entry)

[6] `http://swtch.com/plan9port/man/man1/mk.html`(Plan 9 mk man page)