Truly Universal Binary

Mitchell Olsthoorn

March 26, 2019

1 Introduction

evolution of code modularization debian > plugins > websites > micro services > smart contracts

———

Over the years, the way we use code has evolved with the changing need of the users and the society as a whole. This evolution started off with specific applications written for each use case and each platform it had to run on. These application took a lot of time to develop and could not be reused. To reduce this time, abstraction libraries were built to make it possible to run these applications on similar platforms. These abstraction layers, however, were still limited to broader types of platforms e.g. Linux, Unix, Windows, Mac. The platform libraries could now be maintained and distributed separately. This led to easier development and application that could be used on more systems.

The Debian package system is a good example of the beginning of this evolution. It made it possible for code that was meant to be used as a library to be packaged separately for both system and user code. This allowed applications to indicate which library would be requirement for the application and the system would make sure it is available to it. This possibility allowed applications to be developed even faster.

These new code libraries provided a lot of benefit and speed to application developers, but to improve the ecosystem further a new step had to be made. At this point when applications were distributed they were static. The was no option to adapt the application to include features that the user would like. Also users that wanted to add there own functionality had to go through the developers to accomplish this. To solve this, larger application began to include plugin systems. A plugin system allows different parts of the code to be changed or to add functionality to the application. This paradigm allowed rapid development of extra features by both developers and the users of the application.

A very early example of a program with a plugin system is Winamp. The Winamp developers used the plugin system to provide users with a customisable package that could serve each user's preference. A large community formed around the application with different plugins for every imaginable feature. This was the start of the plugin community.

When the whole application movement started to go to the web, this same plugin paradigm started to exist. These plugins allowed external parties to add functionalities to some of the biggest websites. A good example of this is Facebook plugins. Even now when Facebook is in a decline, people still actively use and rely on plugins hosted on Facebook.

This modularization continued when web application started to use the micro-services architecture. This allowed web application to move towards modules that had very small tasks that they were specifically designed for. This facilitated code reuse on a big scale with platforms like NPM and reusable web components.

The decentralised application community eventually also started to work on modular applications in the form of smart contracts. Ethereum is a good example of this.

this thesis presents a new paradigm for trustworthy computing. We build on the large body of work around smart contracts and make it more generic, scalable, removed global consensus, and need for oracles. It represents the next step in the continued evolution of computing models.