Vim is a powerful text editor with a big community that is constantly growing. Even though the editor is about two decades old, people still extend and want to improve it, mostly using Vimscript or one of the supported scripting languages.

Motivation

Over its more than 20 years of life, Vim has accumulated about 300k lines of scary C89 code that very few people understand or have the guts to mess with.

Another issue is that as the only person responsible for maintaining Vim's big codebase, Bram Moolenaar, has to be extra careful before accepting patches, because, once merged, the new code will be his responsibility.

These problems make it very difficult to have new features and bug fixes merged into the core. Vim just can't keep up with the development speed of its plugin ecosystem.

Solution

Neovim is a project that seeks to aggressively refactor Vim source code in order to achieve the following goals:

- Simplify maintenance to improve the speed that bug fixes and features get merged.
- Split the work among multiple developers.
- Enable the implementation of new/modern user interfaces without any modifications to the core source.
- Improve the extensibility power with a new plugin architecture based on coprocesses. Plugins will be written in any programming language without any explicit support from the editor.

By achieving those goals new developers will soon join the community, consequently improving the editor for all users.

It is important to emphasize that **this is not a project to rewrite Vim from scratch** or transform it into an IDE (though the new features provided will enable IDE-like distributions of the editor). The changes implemented here should have little impact on Vim's editing model or Vimscript in general. Most Vimscript plugins should continue to work normally.

The following topics contain brief explanations of the major changes (and motivations) that will be performed in the first iteration.

Migrate to a CMake-based build

The source tree has dozens (if not hundreds) of files dedicated to building Vim on various platforms with different configurations, and many of these files look abandoned or outdated. Most users don't care about selecting individual features and just compile using `--with-features=huge`, which still generates an executable that is small enough even for lightweight systems by today's standards.

All those files will be removed and Vim will be built using CMake, a modern build system that generates build scripts for the most relevant platforms.

Legacy support and compile-time features

Vim has a significant amount of code dedicated to supporting legacy systems and compilers. All that code increases the maintenance burden and will be removed.

Most optional features will no longer be optional (see above), with the exception of some broken and useless features (e.g. NetBeans and Sun WorkShop integration) which will be removed permanently. Vi emulation will also be removed (setting `nocompatible` will be a no-op).

Platform-specific code

Most of the platform-specific code will be removed and libuv will be used to handle system differences.

libuv is a modern multi-platform library with functions to perform common system tasks, and supports most unixes and Windows, so the vast majority of Vim's community will be covered.

New plugin architecture

All code supporting embedded scripting language interpreters will be replaced by a new plugin system that supports extensions written in any programming language. Compatibility layers will be provided for legacy Vim plugins written in some of the currently supported scripting languages such as Python or Ruby. Most plugins should work on Neovim with little modification, if any.

Moreover, GUIs are implemented as plugins, decoupled from the Neovim core.

See the Plugin Architecture page for a detailed overview.