Tulu-C-IDE Helix Edition

Helix

What is Helix?

From their website and GitHub repository, we can find their description of Helix: A post-modern **text editor**.

Helix is a toy version of the revered Vim text editor, with **batteries included**. It is as much a toy as our mascot Tulu. He (Tulu) looks like a toy, but a close inspection will reveal that he is a real thing. Helix is on the same page.

Helix is a **no-nonsense** text editor inspired by Kakoune, Vim, and Neovim written from scratch in RUST. Helix provides us with crucial Vim features we actually need and tasks we often perform in Vim, so less configuration is needed. The only difference is that it has a slightly different way of working. Open a terminal of your choice (e.g., CMD.EXE) and type `hx`. Go to Command Mode `SHIFT` + `;` (i.e., `:`) .

https://user-images.githubusercontent.com/16861933/102935218-baebd680-44cb-11eb-996c-f92435a903c1.gif

Then type `tutor`.

Is there a downside? Yes and No. If you expect Helix to be a text editor with a GUI and Menubar like Geany, VSCode and Notepad++, you are out of luck. It's quite similar to Neovim in this regard. However, Helix won't be difficult to use and you won't experience any inconvenience either. Nothing to worry about at all!

Batteries Included: Helix comes with built-in LSP (Language Server Protocols) support for autocompletion and syntax checking without any external plugin or extra configuration. Most common needs are served out of the box. It is pretty much an Install-n-Go text editor.

Installation on Microsoft Windows 10 (x64) and Ubuntu/Ubuntu Derivative Linux:

Their official website is: https://helix-editor.com/

You'll have to install MSYS2 (x64) and 64-bit versions of RUSTC and CARGO.

Look here.

The Helix installation part:

Open the Microsoft Windows Command Prompt.

`WINDOWS` + `r` -> `cmd`

Ubuntu:

Open any terminal emulator you prefer. `CTRL` + `ALT` + `T` (Ubuntu XFCE or Ubuntu GNOME).

Navigate to a directory where you have at least 4 GB of drive space left.

Clone the repository.

```
git clone https://github.com/helix-editor/helix
```

Enter the directory containing the source files of 'helix'.

```
cd helix
```

```
git pull
```

You'll have to add some folders from that downloaded source code directory to your system environment before you can compile the program. Since you have already downloaded the source into a directory on your system, you'll have to find the required directories from there.

Say you have downloaded the code into `D:\helix` or, `%userprofile%\some_source_to_build\helix` or, `$HOME/helix`. In that case, temporarily set your environment variable in your Windows Terminal, here, `cmd.exe`, as `setx HELIX_RUNTIME "D:\helix\runtime"` or `setx HELIX_RUNTIME "%userprofile%\some_source_to_build\helix\runtime"`. On Linux, add a line to either your `.bashrc` or `.bash_aliases` as `HELIX_RUNTIME=/home/YOUR_USERNAME/helix/runtime/` assuming your source is downloaded into `$HOME/helix`. Do you use `musl-libc` instead of `glibc` on a Linux distribution like Alpine? If yes, paste the following command into your Terminal Emulator before compiling the code. If not, skip setting the `RUSTFLAGS` and move on to the `cargo install` step.

```
RUSTFLAGS="-C target-feature=-crt-static"
```

Build & Install Helix.

MS Windows:

Try compiling the code from the command prompt (Windows) first. If that doesn't work, type `msysb` (find it in `msysb.cmd/msysb.cmd`) or simply fire up the MSYS2-x64 console (Blue).

```
cargo install --path helix-term --locked
```

Wait a couple of minutes till the build process is complete. Don't close the Command Prompt. Keep it open.

Specific build instructions for MS Windows:

Open `cmd.exe` with Administrator Privilege. WINDOWS + r -> cmd -> CTRL + SHIFT + ENTER.

create a file `config` in `%USERPROFILE%\.cargo`.

```
notepad %USERPROFILE%\.cargo\config
```

Paste the following lines into the file `config`, reference.

```
[target.x86_64-pc-windows-gnu]
linker = "C:\\msys64\\mingw64\\bin\\gcc.exe"
ar = "C:\\msys64\\mingw64\\bin\\ar.exe"
```

Type `msysb`. Find the MSYS2-x64 configuration for Windows Command Prompt in the `console-config/` directory.

Clone the Helix GIT repository.

```
git clone https://github.com/helix-editor/helix
```

```
cd helix
```

```
exit
```

(The MSYS2-x64 shell is no longer needed.)

In the Command Prompt,

```
cd helix
```

```
setx HELIX_RUNTIME /n/helix
```

```
cd %appdata%\helix mklink /D runtime "N:\helix\runtime"
```

Edit the Helix build configuration:

```
notepad Cargo.toml
```

Add the following lines at the beginning of the file, reference.

```
default_toolchain = [
  "stable-x86_64-pc-windows-gnu"
  ]
```

The build command `cargo install --path helix-term --locked` was expected to work. However, the recent changes to the Helix editor seem to have trouble linking the compiled binary files against MinGW-w64 supplied library files provided by MSYS2. The issue is expected to be resolved in the near future.

Check your GCC version:

```
gcc -v
```

Try each of the following commands successively if one command fails. Ref: Cross compiling issue with static linking - help - The Rust Programming Language Forum

```
cargo install --target="x86_64-pc-windows-gnu" --path helix-term --locked
```

```
cargo install --target="x86_64-pc-msys" --path helix-term --locked
```

```
cargo install --target="x86_64-w64-mingw32" --path helix-term --locked
```

Helix will be installed into `%USERPROFILE%\.cargo\bin` (Linux: `~/.cargo/bin`). The name of the Helix executable is `hx.exe`. On Linux: `~/.cargo/bin/hx`.

`C:\Users\YOUR_USERNAME\.cargo\bin\hx.exe`.

Add `%USERPROFILE%\.cargo\bin` to System PATH (Environment Variables).

Helix will look for its configuration file `config.toml` (.vimrc/init.lua equivalent) and LSP query files in `%AppData%\helix` (Linux: `~/.config/helix/`). The folder `%AppData%\helix` (Linux: `~/.config/helix/`) is empty as of now. We will have to provide Helix with the files it requires.

The cloned repository contains a folder `runtime`. We will have to copy this folder to `%AppData%\helix` (Linux: `~/.config/helix/`).

In the Command Prompt, type:

```
xcopy /e /i runtime %AppData%\helix\runtime
```

Optionally,

```
hx --grammar fetch
```

```
hx --grammar build
```

Helix will look for 'theme' files in `%AppData%\helix\themes` (Linux: `~/.config/helix/themes`). The cloned repository comes with a plethora of themes. The folder `runtime` contains all the themes that were also copied along with the `runtime` files. We will have to copy the `themes` folder to `%AppData%\helix` from `%AppData%\helix\runtime` (Linux: from `~/.config/helix/runtime` to `~/.config/helix`).

```
xcopy /e /i %AppData%\helix\runtime\themes %AppData%\helix\themes
```

The best way to get Helix work on MS Windows is to grab the latest build from Helix's GitHub repository.

Helix's GitHub repository: https://github.com/helix-editor/helix/releases

Extract the ZIP archive.

Open the Windows File Manager. Type `%APPDATA%\helix` at the address bar and enter `%APPDATA%\helix`. Copy the folders `contrib` and `runtime` from the extracted ZIP archive to `%APPDATA%\helix`.

Enter `%USERPROFILE%\.cargo\bin\` from the GUI File Manager as well. Copy `hx.exe` from the extracted archive to that folder.

If any of the folders (either `%APPDATA%\helix` or `%USERPROFILE%\.cargo\bin\`) does not exist on your system, create the same using the `mkdir` command. Example: `mkdir %USERPROFILE%\.cargo\bin\`.

If Helix has been built without any trouble on your MS Windows system, follow the steps mentioned below.

Ubuntu:

```
mkdir ~/.config/helix/runtime
```

```
cp -r $PWD/runtime/ ~/.config/helix/
cp -r ~/.config/helix/runtime/themes ~/.config/helix/themes
```

Helix configuration file

Open the supplied **Helix configuration file** `config.toml` (inside `_helix_edition`) with a GUI text editor like Notepad2-mod. **Read it line by line**. Everything is pretty self-explanatory. Change settings as you wish.

Now, copy this `config.toml` file to `%AppData%\helix` (Linux: `~/.config/helix/`) folder. You can drop this file into the cloned repository first and then use the Windows terminal/(Any Linux terminal emulator) to copy the file as well.

```
copy config.toml %AppData%\helix
```

Ubuntu:

```
cp $PWD/config.toml ~/.config/helix/
```

NOTE for Windows Users: The folder `grammars` that contains *Tree-sitter parsers* (syntax highlighting etc.) is around 770 MB, and this directory contains several Dynamic Link Library (Windows shared library) files along with their sources. If you find that the Windows file copier failed to copy this directory properly at the first attempt, you'll have to take one extra step.

```
cd runtime\
```

```
xcopy /e /i grammars %AppData%\helix\runtime\grammars
```

Finally, visit the directory `%AppData%\helix\runtime` and check whether everything is okay. Without the Tree-sitter parser files, you'll miss syntax highlighting and many features of Helix.

Find the location where the Helix executable is installed. The `where` command will tell you the location.

```
where hx
```

```
C:\Users\YOUR_USERNAME\.cargo\bin\hx.exe
```

Means,

```
%USERPROFILE%\.cargo\bin\hx.exe
```

Ubuntu:

```
whereis hx
```

Run Helix health check to see whether everything is set up as expected:

```
hx --health
```

Helix will show the LSPs installed.

```
Language            LSP                 DAP                 Highlight
Textobject          Indent
c                   ✓ clangd            ✓ lldb-vscode        ✓
✓                    ✓
cpp                 ✓ clangd            ✓ lldb-vscode        ✓
✓                    ✓
opencl              ✓ clangd            None                ✓
✓                    ✓
rust                ✓ rust-analyzer     ✓ lldb-vscode        ✓
✓                    ✓
zig                 ✘ zls               ✓ lldb-vscode        ✓
✓                    ✓
```

Alternative Installation Method:

In a nutshell, Helix requires the following files in their respective directories as follows:

Executable: `hx.exe` -> `%USERPROFILE%\.cargo\bin` [Linux: `~/.cargo/bin`]

Folder: `runtime` -> `%AppData%\helix\runtime` [Linux: `~/.config/helix/runtime`]

The folder `runtime` contains three subfolders and one text file:

`grammars`

`queries`

`themes`

`tutor` (file)

Custom configuration file: `config.toml` -> `%AppData%\helix` [Linux: `~/.config/helix/`]

Helix offers precompiled binary distributions of the editor on their GitHub Release page. Download Helix from there and extract the archive, after which you can copy the folders/files in the respective directories on your system.

On Linux, choose either the Flatpak version or the AppImage package.

Flatpak:

```
flatpak install flathub com.helix_editor.Helix
flatpak run com.helix_editor.Helix
```

AppImage:

```
chmod +x helix-*.AppImage #change permission parameters
```

Create a Symlink if necessary:

[Delete the old symlink while creating a new one for a newer version of Helix.]

```
rm ~/.local/bin/hx
```

```
ln -s ~/PortablePrograms/helix-*.AppImage ~/.local/bin/hx
```

A detailed build instruction can be found here: Installation.

How will you use Helix:

1. **RUST**: You need a **Cargo-generated project folder** to work with RUST source files. `rustc` , `cargo` , and `rust-analyzer` must be found on the System Search PATH. MSYS2 usually installs the complete RUST toolchain including the `rust-analyzer` .

```
rust-analyzer --version
```

`rust-analyzer 1.64.0 (8517eff67 2022-09-23)`.

If you don't see it on the system, find the file `rust-analyzer-build-instructions.txt` and follow the instructions.

2. **C/C++**: You need a `compile_commands.json` or a `compile_flags.txt` file alongside the code being edited. `clangd` along with the entire LLVM Clang toolchain must be found on the System Search PATH.

3. Other Languages: **I'm not sure.** Have a look at the supported LSPs here: Language Support

Open CMD.EXE and navigate to an appropriate directory containing source codes, and type `hx` `<space>` `filename_with_extension`. For example:

```
hx main.rs
```

More about build and install:

https://github.com/helix-editor/helix#installation

If you want to learn and use Helix, go ahead and do so. Helix is a complemental utility to GVim. You'll need both on different occasions. In a terminal-only environment where a GUI is inaccessible Helix works better without any hassle of setting things up. For a quick edit, Helix is a better choice. Helix also looks better in terminals. On the other hand, GVim can be extended beyond imagination. They are complemental to one another.

Copyright Notice: