GSoC Proposal

About Me

Basic Information

- Name: Fan Chung
- Major: Computer Science
- University: National Yang Ming Chiao Tung University, Taiwan
- Email: thesummernightcat@gmail.com
- Timezone: Asia/Taipei (UTC +8)
- Github: https://github.com/cycatz

Brief Bio

I am Fan Chung, a sophomore at National Yang Ming Chaio Tung University. I'm pursuing a bachelor's degree in Computer Science.

I am proficient in C/C++, Shell Scripts, and Python. I am familiar with the Unix environment, as I use Archlinux and FreeBSD on my computers to complete most of the tasks in my daily life.

As a native speaker of Chinese, I have some basic knowledge of Chinese writing system, including character encoding, CJK fonts, and input method. And I had experience working on these projects:

- InconsolataLGC-TaipeiSansTC[1] : A hybrid font merged from *Inconsolata LGC* and *Taipei Sans TC*.
- Cangzen[2] : A website lets you practice *Cangjie*[3] (a Chinese input method), including key positions and decomposition rules.
- chemical-kb[4]: A custom *Rime* [5] input schema that helps you input chemical element characters in Chinese fast and easily.

Introduction

Idea

Input method in FreeBSD virtual terminal

Currently, the FreeBSD virtual terminal driver `vt(4)` does not support inputting CJK characters directly in the virtual terminal, so our project idea is to provide an environment that can run IME (input method editor) in the console.

Background

A common usage scenario is when a user wants to create a user with the user info description in Chinese during the system installation. Since the installation process must proceed in a virtual console, there is no way to open an IME window to input CJK characters.

Another example is hosting a FreeBSD server on a cloud platform or a virtual machine. When one wants to do some simple admin routines on the machine or disconnects from the SSH session, he/she needs to do some configurations and keeps records or takes notes on the server. It's sometimes quite inconvenient for those CJK writing-system users to write texts in their most familiar languages.

[1] https://github.com/Cycatz/InconsolataLGC-TaipeiSansTC
[2] https://github.com/Cycatz/Cangzen
[3] https://en.wikipedia.org/wiki/Cangjie_input_method
[4] https://github.com/Cycatz/chemical-kb
[5] https://rime.im/

This project can be divided into two parts, I call them backend and frontend:

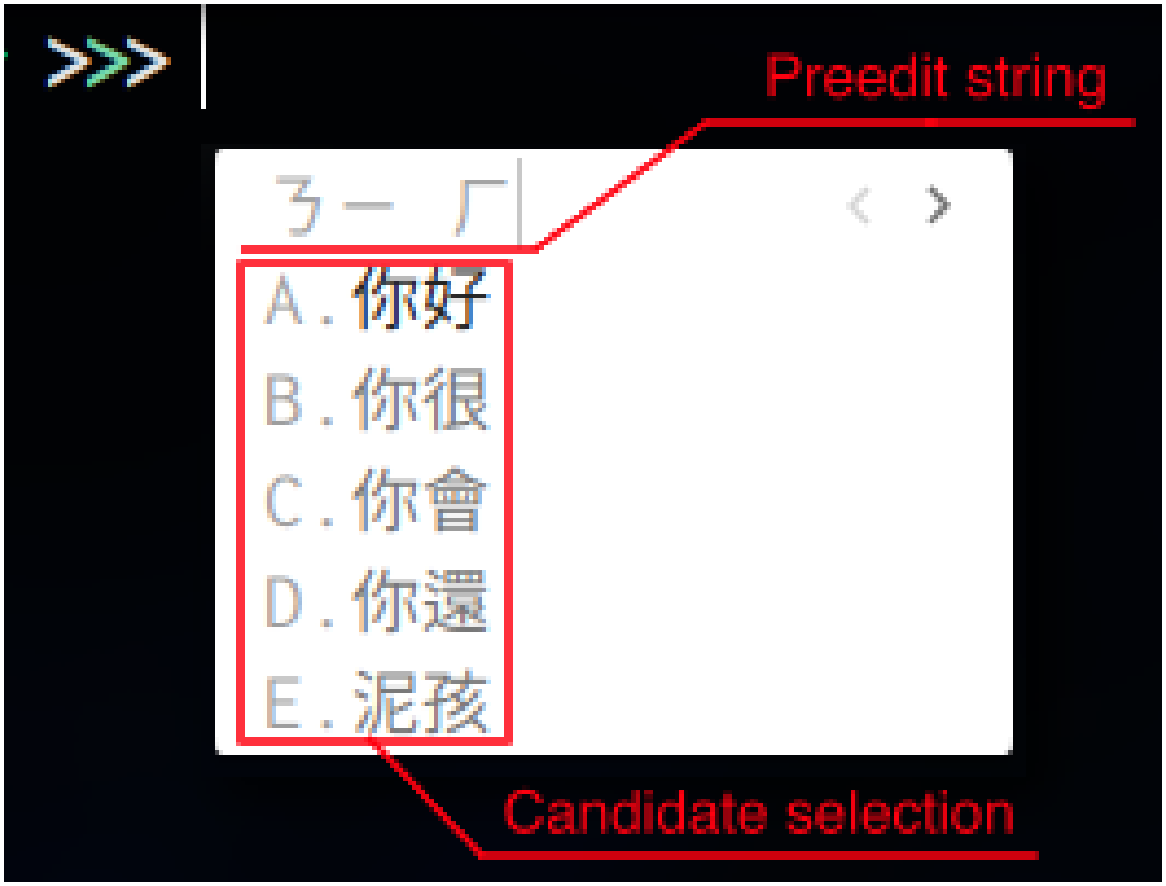
Backend

The backend needs to handle raw keystrokes passed from the frontend, process some special keys, and send ordinary keys to the IME library. After the IME library translates the keystrokes and returns a valid CJK character, then the backend will pass the result to the frontend.

Frontend

The frontend is responsible for drawing the IME window on the screen. The IME window is mainly composed of two parts:

- Preedit string: The preedit string contains temporary characters which are not typed yet on screen. They might be already translated or not be translated into final CJK characters.
- Candidate selection: Some input methods may produce multiple possible results when translating keystrokes into a valid character, we call those results "candidate". The user has to choose and select the most suitable candidate he/she wants.



And we want to implement similar features of those IME (fcitx, ibus) but in the console. There are two ways to achieve this goal, one is implementing in pure text mode, another is in graphic mode.

Text Mode

The text-mode way only allows us to draw the screen with characters rather than individual pixels. Therefore we could use terminal escape characters to control and draw graphics on the screen, or use 3rd library like *ncurses*[6]. Moreover, a easy-implemented idea is to print IME information on the `tmux` status line. With the flexible and customizable `tmux` configuration, we can easily provide a user-friendly interface to show the components of the preedit string and candidate selection.

[6] https://invisible-island.net/ncurses/announce.html

Graphic Mode

In graphic mode, we can directly manipulate individual pixels, so it's more flexible than in text mode in terms of drawing graphics. Although some existing virtual consoles do support CJK display and input methods, e.g., zhcon[7], big5con. However, these tools do not support utf-8 very well[8] and only provide a few input methods. Therefore we want to take those features and bring them into `vt(4)` with improvements like utf-8 full support and integration with modern input method engine, e.g., *Rime*.

Project Goals & Implementation

- Create an IME backend API that accepts keystrokes as arguments or messages and communicates with the input method engine library.
  - Use `librime`[9] as the input method engine library to translate keystrokes into valid CJK characters.
  - Provide extra features like schema selection, deployment, and data synchronization.
- Create a graphical frontend to be able to show the current input state and candidates of CJK characters.
  - Test stage: Print the IME information on the `tmux` status line. You may ask why `tmux`? Because it's more convenient for us to draft the UI at an early stage and decide how to arrange and display those IME components on the screen. Furthermore, `tmux` let us customize keybindings and call external scripts easily.
  - Final stage: Display IME components on the virtual terminal console. And we also need to handle those keystrokes manually and send them to the backend.

Timeline

Community Bonding Period (May 17 ~ June 6)

- Get familiar with `librime` code base
- Read `vt` source code[10]
- Also survey those projects which use `librime`, e.g., `emacs-rime`[11]

Week1 ~ Week2 (June 7 ~ June 21)

- Interact with mentors.
- Set up a skeleton for the IME backend API.

Week3 ~ Week4 (June 22 ~ July 7)

- Fully Implement the backend functions.
- Implement UI with `tmux` and connect it with the backend API.
- Perform tests and fix bugs.

Week5 (June 7 ~ July 15)

- Finish IME in text mode.
- Period for any unexpected delay.

Mid term Evaluation

[7] http://zhcon.sourceforge.net/index_cn.html
[8] https://github.com/Lian0123/twcall
[9] https://github.com/rime/librime
[10] https://github.com/freebsd/freebsd-src/tree/main/sys/dev/vt
[11] https://github.com/DogLooksGood/emacs-rime

Week6 ~ Week7 (July 16 ~ August 1)

- Get familiar with `vt` code base again.
- Implement the rough structure of the IME UI in graphical mode.

Week8 ~ Week9 (August 2 ~ August 15)

- Fully Implement the display part and the typing part in graphical mode.
- Perform tests and fix bugs.

Final Week (August 16 ~ August 23)

- Finish IME in `vt` graphical mode.
- Final testing and debugging of the project.
- Period for any unexpected delay.