

nIME - A better Japanese IME for plan 9 and 9front

Ethan Long

Email 1: ethandavidlong@gmail.com

Email 2: u7281759@anu.edu.au

Phone: +61 476 155 407

Chat: ethan on 9p.zone or lightningx10 on freenode

ABSTRACT

nIME (short for nine IME, pronounced nie-em-ee or nine-em-ee) is a proposed IME for plan 9's `rio` environment that more closely replicates the modern IMEs on other operating systems such as `anthy`, `fcitx`, or `google's IME`.

Author Background

I am a first year university student doing a Bachelor of Science with Honours (ASCAD) at the Australian National University in Canberra, Australia. I am currently studying Mathematics, Physics, Computer Science, and Chemistry. Additionally I am part of the avionics team of the ANU Rocketry Society, we are a club that participates in various rocketry competitions and we are currently planning and building a rocket to reach the karmen line. I have no formal training in C, but I have worked with plan 9 and C on other operating systems, just not on a kernel/operating systems programming level.

I have been using 9front as a development/working environment for over a year, currently opting for a `vmm` instance that I drawterm into. Because of this, I have good knowledge of how the operating system works from a usage and operational perspective, I.E. how it all fits together. I have yet to get into extensive programming on plan9 apart from basic GUI programs, so the project proposal was picked to reflect what should be a relatively simple backend.

I have been studying Japanese since 2015, I did a second year University course on Japanese last year as an extension to my high school studies. Therefore whilst not native in Japanese, I am able to work with the language effectively in an informal context.

The only code samples of my work available will be on my gitlab, most of my work is done with the school, so I am unable to share a lot of it for academic integrity reasons. The gitlab is available at <https://gitlab.com/lightningx10/>

I edited the p9f wiki sandbox to include the statement "ethan was here" on Apr 13 07:54:59 PDT.

Project Information

Despite having UTF-8 support, the current solutions for inputting Japanese into plan 9 are old and inadequate for most modern usage, with `ktrans` being the best implementation available. `ktrans`, despite having fairly good coverage of the Kanji, is not intuitive or convenient to use. If one wants a particular Kanji and they accidentally miss it in the list of suggestions, they have to delete what they typed and retype it. Furthermore, if a Kanji is not available in the list of predefined Kanji, there is no easy way of telling apart from cycling the list multiple times, which can take a long time. Typing in `alsoktransis` which

is rarely used nowadays due to its complexity to new users.

The solution to this is to create a new IME for plan 9 using similar principles to `ktrans`, but with a graphical kana to kanji conversion context menu that uses either the keyboard or the mouse to select the desired conversion. The user will likely still need to convert words individually rather than whole sentences (it's a lot more complicated to implement such a system), but doing so will become easier.

For reference on the kind of menu that this project desires to replicate, see the following graphic:

Input Word in hiragana
Conversion 1
Conversion 2
Conversion 3
Conversion 4
Conversion 5
...

(Ghostscript does not let the user input Japanese UTF text when using standard troff output, so the example is in English)

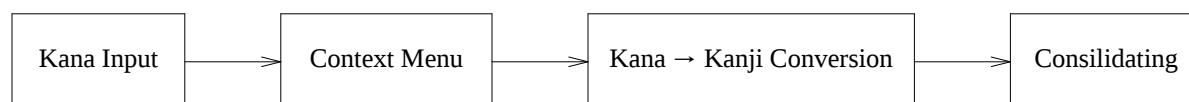
The list of suggestions would appear if the user is in Hiragana input mode and hits the Space key, as that has no grammatical use in Japanese and is a convenient switch to hit. The list would be cycled through using Space and Shift+Space to go down and up respectively, and also by use of the mouse pointer. Like `ktrans`, `nIME` would have separate Hiragana, Katakana, and English input modes, with the Hiragana input mode offering the additional Kana to Kanji conversion.

In order to get the list of conversions, a similar method to `ktrans` would be employed, having a dictionary for kana like the SKK one distributed with `ktrans`. Such dictionaries are now widely available and open source, but feature parity with `ktrans` could be met by simply re-using `ktrans`' pre-existing supplied open source dictionary.

Longer term, supplemental programs to add new words to the dictionary could be made. The program should also be futureproof, it should be as modular and hackable as possible to allow for future improvements in giving more personalised conversions, or full sentence conversions.

Approximate Schedule

An approximate order of implementation:



Kana Input: Kana input could conceivably take up to two weeks to implement, as it would need to include most of the code that intercepts the keyboard and inserts input into programs.

Context Menu: The context menu won't take long to implement, `rio` already makes this easy, the hardest part will be getting it to accurately place down and getting input from the keyboard as well as the mouse. The estimated time for completion thus would be around a week at most.

Kana -> Kanji Conversion: This step will by far be the longest. A language input system should feel like it is native in the language that it is taking, I.E. the system should feel like a Japanese person would actually use it for input. The first step would be deciding on the dictionary system to use, whether to use `ktrans`'

existing SKK dictionary, or to look for a new one. The dictionary would be implemented in plain text, either in CSV, or in TSV like the one `ktrans` uses. Next, the reading of the dictionary would be implemented, along with perhaps some kind of weighting system to ensure that the more common Kanji conversions appear at the top of the list. Then finally testing of the program to make sure it is as functional and seamless as possible for a summer project. Thus this would be roughly around 5-6 weeks of work.

Consolidation: If the project is satisfactory in its the implementation and there is remaining time in the project after the last step, then additional features would be implemented. Such features may include personalised dictionary helper programs, ways of personalising the weighting of Kanji suggestions to allow for faster typing, conversion of full sentences (Identifying grammar structures).

Availability

During the first week of the GSoC, the ANU will still be holding exams. No lectures will be held during this time, and I have no other commitments outside of the ANU Rocketry Society. My biggest drop in availability will be during the final 3 weeks of the GSoC, when the second semester starts at the ANU. I am therefore planning on squeezing the bulk of the work on this project into the first 8 weeks of the GSoC timeline.

Mentor Communication

For quick back and forth communication, I would be most happy communicating with my mentor over the 9p.zone chat server hosted by kvik, but if the mentor wants to talk over the 9fans GSoC IRC channel that would also be fine. For updates and long term status communication, I would be most happy using email to communicate with them. If my mentor disappeared for any reason, I'd continue to provide them and the 9fans GSoC IRC with updates, whilst asking the wider 9p.zone community for advice if I was really stuck.