VT-IME: Input Method Editor in FreeBSD vt(4)

BSDCan 2023

Fan Chung (Cycatz)

May 20, 2023

Outline

Background

VT-IME

Demo

Future work

Q & A

About me

```
$ id cycatz
```

- Fan Chung
- Undergraduate @ NYCU, Taiwan
- Exchange student @ UIUC, US
- Been using FreeBSD for ~3 yrs
- Like *nix ricing

Background

Input Method Editor (IME)

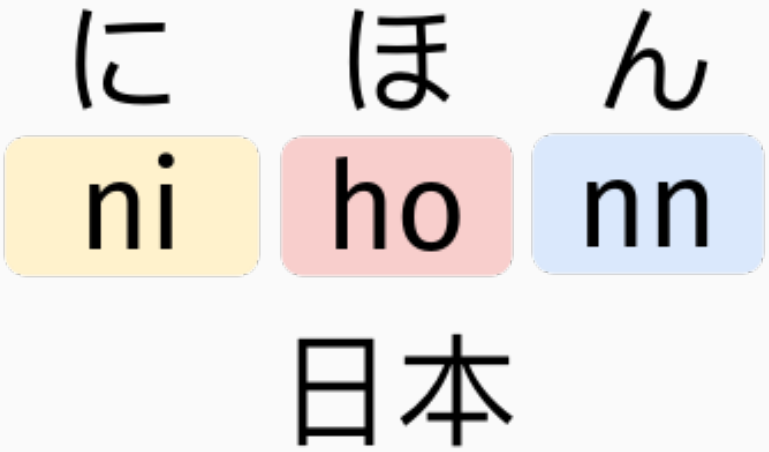
- Input characters that can't be typed on standard keyboard
  - Over **3000** characters are required in daily Chinese writings
- `chinese/fcitx`, `textproc/ibus`

Chinese keyboards, huh?

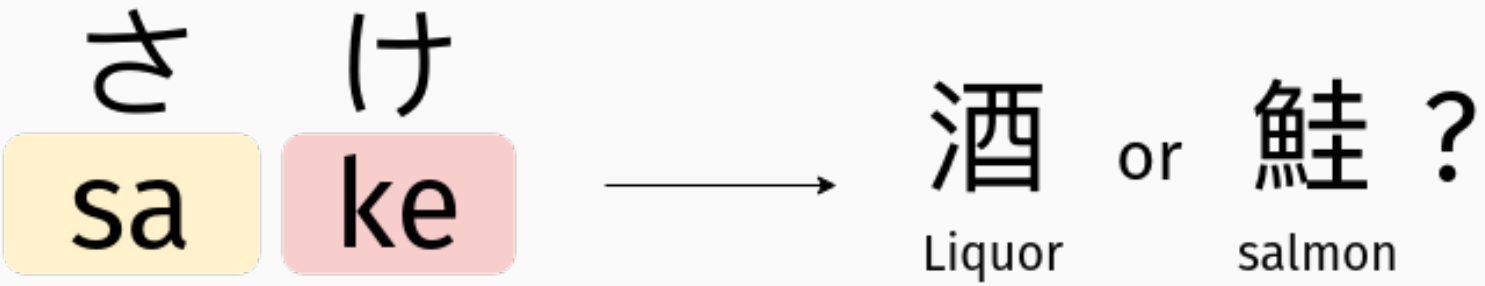
Input Method [Engine]

- An IME typically support many *input method engines*, which implement one or more *input method* (IM)
- *Input method* (IM) is a set of algorithms to convert keystrokes into characters

Input Method [Engine]



Input Method - Example



→ The user have to choose the word they want to type!

Hierarchy of Input Method [Editor|Engine]

- Input method editor
  - Input method engine 1
    - Input method a
    - ...
  - Input method engine 2
    - ...

Hierarchy of Input Method [Editor|Engine]

- chinese/fcitx
  - chinese/fcitx-rime
    - chinese/rime-bopomofo
    - ...
  - japanese/fcitx-mozc
  - ...

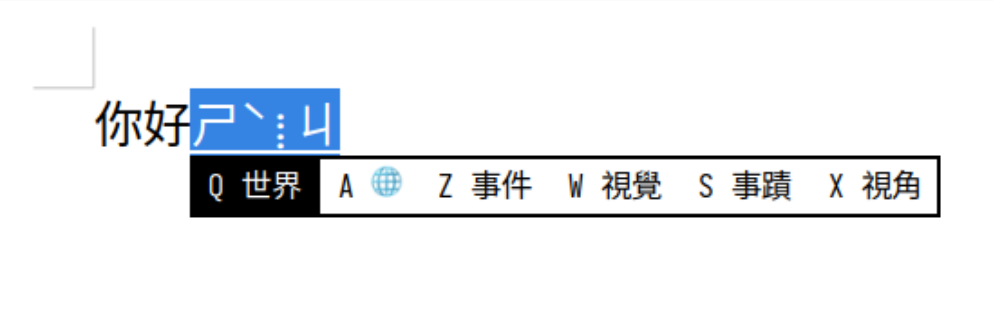


Figure 1: Fcitx, an input method editor

CJK

- *CJK* is an acronym for “Chinese, Japanese, and Korean”
- *CJK character* refers to the Chinese characters in the writing systems of the three langauges

Virtual Terminal in FreeBSD

- Two implementations: `syscons(4)` and `vt(4)`
- `vt(4)` is the newer implementation featuring:
  - UTF-8 encoding
  - double-width characters

  $\rightarrow$ Make it possible to display CJK characters!

Why we need a Input Method Editor in FreeBSD terminal?

- Still cannot input CJK characters directly in `vt(4)` !
  - For example, *fcitx*, depends on GUI framework, such as X
- Unrealistic to install a whole GUI framework for simple text processing

VT-IME

VT-IME — Introduction

- **VT-IME** — a system integrating an IME into vt(4)
- Able to type CJK characters w/o IME & GUI framework

VT-IME — Components

VT-IME consists of two parts: frontend & backend

Frontend — patched `vt(4)`

- Capture key events and render the IME interface

Backend — an userspace server

- Run an *IM server* in the userspace to translate key press events to CJK characters

VT-IME — Diagram

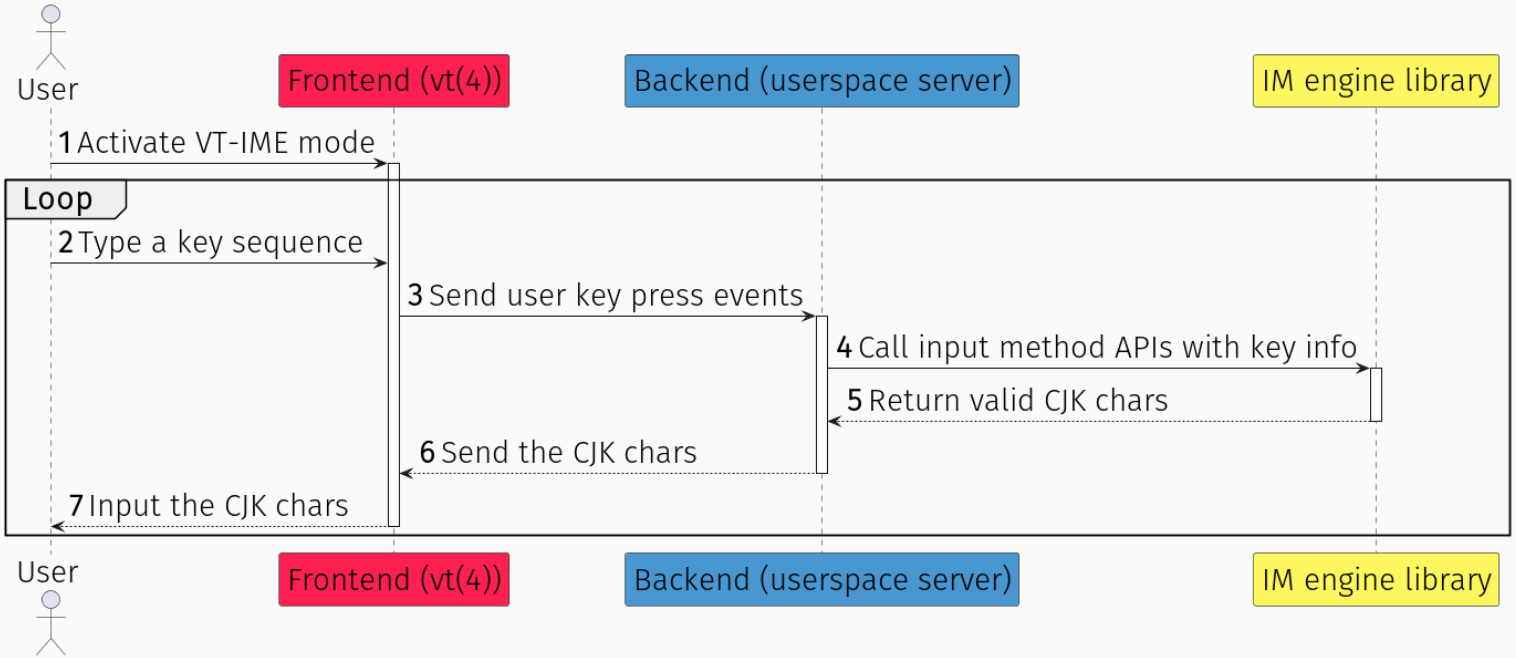


Figure 2: VT-IME diagram

VT-IME — Message Types

1. *key* — send visible key information
    - e.g., `'a'`, `'3'`
2. *raw* — send special key information
    - e.g., `Enter`, `Space`
3. *delete* — request the deletion of a character
4. *output* — request the current status text
5. *exit* — quit VT-IME mode

VT-IME — Frontend

- Hack into `vt(4)` code
  - `src/sys/dev/vt`
- Q: Where does `vt(4)` receive key events?
  → A: `vt_processkey()`
- Q: Where does `vt(4)` input chars?
  → A: `terminal_input_(char|special|raw)`
- Approach:
  - Intercept `vt_processkey()`
  - Call `terminal_input_(char|special|raw)`

VT-IME — Frontend

```
1  void
2  vt_processkey(keyboard_t *kbd, struct vt_device *vd, int
   ↪  c)
3  {
4       /* ... */
5
6  #if VT_IME
7         if (vt_ime_is_enabled(&vt_ime_default))
8           vt_ime_process_char(vw->vw_terminal,
9                               main_vd,
10                              &vt_ime_default,
11                              KEYCHAR(c));
12        else
13 #endif
14          terminal_input_char(vw->vw_terminal,
15                              KEYCHAR(c));
16      } else
17        terminal_input_raw(vw->vw_terminal, c);
18      /* ... */
19 }
20
```

VT-IME — Frontend

```
1  void
2  vt_ime_draw_status_bar(struct vt_device *vd, char *status)
3  {
4    /* ... */
5    term_char_t ch = FG_WHITE | BG_BLUE;
6    int len = strlen(status);
7    while (len-- > 0) {
8      ret = vt_ime_convert_utf8_byte(&utf8_left,
9                                     &utf8_partial,
10                                    *c++);
11     if (ret <= 0)
12       continue;
13     vb->vb_ime_buffer[blen++] = ch | utf8_partial;
14     vb->vb_ime_buffer[blen++] = ch
15                               | utf8_partial
16                               | TFORMAT(TF_CJK_RIGHT);
17   }
18   /* ... */
19 }
```

VT-IME — Frontend

```
1  inline term_char_t
2  VTBUF_GET_FIELD(const struct vt_buf *vb, int r, int c)
3  {
4    if (r == 0 && vt_ime_buf_state)
5      return vb->vb_ime_buffer[c];
6    else
7      return ((vb)->vb_rows[((vb)->vb_roffset + (r)) %
8              VTBUF_MAX_HEIGHT(vb)][(c)]);
9  }
```

VT-IME — Backend

- Translate key press events into valid CJK characters
- Choose *librime* as the input method engine library
  - And select the input method *rime-bopomofo*

VT-IME — Screenshot

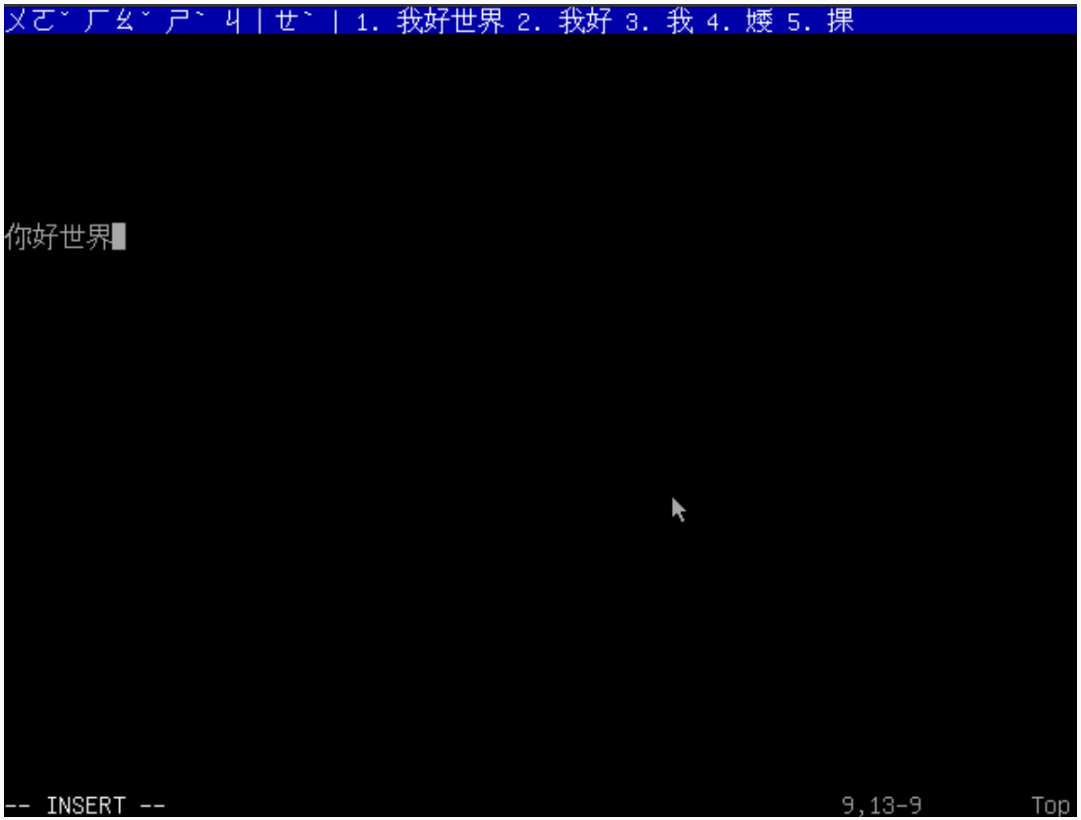


Figure 3: VT-IME screenshot

Demo

Future work

vt(4)

- Need a better pixel font
  - unifont is good, but it's GPL-licensed
- Plane 1 emojis doesn't work
  - Need further investigation
- Handle double-width unicode symbols
  - vt(4) hard-coded only CJK characters

VT-IME

- Better architecture
  - e.g., multiple users
- Support more IME features
  - e.g., input method selection
- Support IME interface customization
  - e.g., customizable key bindings, status bar fg/bg color

Q & A