Japanese Desktop

Christian Külker

2024-06-24

Contents

1 Introduction

This document covers the installation and configuration of a Japanese desktop environment on Debian for different Debian releases. Since the Japanese environment has undergone significant changed in the last 20 years, the installation methods and results are not comparable. The focus of this guide is primarily on a minimal approach and secondarily on the automation of deployment.

2 Debian 12 Bookworm [0.1.8]

2.1 Prerequisites

A running instance of Debian 12 “Bookworm”. Sudo or root access. Basic familiarity with Linux command-line interface.

3 Install Japanese Desktop

```
aptitude update
aptitude safe-upgrade
```

3.1 Information

To understand in which condition the system is the locale and the environment variables are good indicators. For Gnome related desktops like Gnome or MATE ‘gsettings’ should also be taken into considerations.

3.1.1 Locale

While setting the locale **can** solve some problems, it is better to understand the influence of the locale and set only the parts needed. A fully localized desktop have all set to Japanese and will display Japanese everywhere where it is possible. However people who just want to use Japanese, but not live Japanese should not set all variables to Japanese.

```
localectl
System Locale: LANG=en_US.UTF-8
               LANGUAGE=en_US:en
    VC Keymap: (unset)
   X11 Layout: us
    X11 Model: pc105
```

Changing the locale

```
localectl set-locale LANG=ja_JP.UTF-8
localectl
System Locale: LANG=ja_JP.UTF-8
    VC Keymap: jp106
   X11 Layout: jp
    X11 Model: jp106
  X11 Options: terminate:ctrl_alt_bksp
```

3.1.2 Environment

```

QT_IM_MODULE=uim
GTK_IM_MODULE=uim
CLUTTER_IM_MODULE=xim
XMODIFIERS=@im=uim
```

3.1.3 gsettings

```
gsettings list-keys org.gnome.desktop.input-sources

current
mru-sources
per-window
show-all-sources
sources
xkb-options
```

```
gsettings get org.gnome.desktop.input-sources sources
@a(ss) []
```

Should it show [(‘xkb’, ‘us’), (‘ibus’, ‘anthy’)]?

3.2 Task Japanese

The major component is the `task-japanese`:

```
aptitude show task-japanese
Package: task-japanese
Version: 3.73
State: installed
Automatically installed: no
Priority: optional
Section: tasks
Maintainer: Kenshi Muto <kmuto@debian.org>
Architecture: all
Uncompressed Size: 6,144
Depends: tasksel (= 3.73), manpages-ja, lv
Recommends: fbterm, unifont, nkf, manpages-ja-dev
Description: Japanese environment
 This task installs packages that make it easier for Japanese speakers to
  ↪  use
 Debian.
```

It recommends fbterm and unifont.

3.3 Ansible

This is an example of installing and configuring a Japanese desktop via an Ansible playbook.

```
- hosts: service_japanese
  gather_facts: no
  become: yes
  vars:
    ns: service_japanese
    meta_packages:
      - task-japanese
      - task-japanese-desktop
    - ibus  # This will have a menu in the menu bar and no floating panel
      - ibus-mozc
      - mozc-utils-gui
  tasks:
    - name: "{{ ns }}: Ensure Japanese locale is generated"
      ansible.builtin.lineinfile:
        path: /etc/locale.gen
```

```
        regexp: '^# ja_JP.UTF-8'
        line: 'ja_JP.UTF-8 UTF-8'
        state: present
      notify: "{{ ns }}: Reconfigure locales"
    - name: "{{ ns }}: Ensure meta packages are installed"
       # task-japanese: (apt install --simulate task-japanese)
       #   fbterm fonts-unifont libx86-1 lv manpages-ja
     #  manpages-ja-dev nkf psf-unifont task-japanese unifont xfonts-
unifont
       # task-japanese-desktop:
       #   anthy anthy-common fonts-ipafont fonts-ipafont-gothic
     #  fonts-ipafont-mincho fonts-vlgothic im-config kasumi libanthy1
       #   libanthyinput0 libgcroots0 libm17n-0 libotf1 libreoffice-
help-ja
        #   libreoffice-l10n-ja libuim-custom2 libuim-data libuim-
scm0 libuim8
      #  m17n-db mozc-data mozc-server mozc-utils-gui task-japanese-
desktop
     #  uim uim-data uim-fep uim-gtk2.0 uim-gtk2.0-immodule uim-gtk3
        #   uim-gtk3-immodule uim-mozc uim-plugins uim-qt5 uim-qt5-
immodule
       #   uim-xim
      package:
        name: "{{ meta_packages }}"
        state: latest
      notify:
        - "{{ ns }}: Configure im"
    - name: "{{ ns }}: Ensure dconf is installed"
      ansible.builtin.package:
        name: dconf-cli
        state: present
    # --- [ S001 ] ----------------------------------------------
-----------
   - name: "{{ ns }}: Change IBus keyboard shortcut for 'Next input method'"
      dconf:
        key: /desktop/ibus/general/hotkey/triggers
        value: "['<Control>space']"
        state: present
      become: yes
      become_user: "{{ user.user }}"
      register: dconf_output
```

```
    - name: "{{ ns }}: Check if a change was made"
      ansible.builtin.debug:
        # Requires: ANSIBLE_STDOUT_CALLBACK=debug
        msg: |
       ********************************************************************
        * IMPORTANT: NEW ibus configuration applied via dconf.        *
        * A restart of X11.org is required to apply the changes.      *
         * Please ensure this is handled appropriately!               *
       ********************************************************************
      when: dconf_output.changed
    # --- [ S002 ] ------------------------------------------------
-----------
    - name: "{{ ns }}: Crate mozc configuration space directory"
      file:
        path: "/home/{{ user.user }}/.config/mozc"
        state: directory
        owner: "{{ user.user }}"
        group: "{{ user.group }}"
        mode: '0750'
   - name: "{{ ns }}: Create mozc configuration space file if not exist"
      stat:
     path: "/home/{{ user.user }}/.config/mozc/ibus_config.textproto"
      register: file_check
   - name: "{{ ns }}: Crate mozc configuration space file if not exist"
      file:
     path: "/home/{{ user.user }}/.config/mozc/ibus_config.textproto"
        state: touch
        owner: "{{ user.user }}"
        group: "{{ user.group }}"
        mode: '0750'
      when: not file_check.stat.exists
    - name: "{{ ns }}: Activate Mozc Hiragana conversion on launch"
      ansible.builtin.lineinfile:
     path: "/home/{{ user.user }}/.config/mozc/ibus_config.textproto"
        regexp: '^active_on_launch: False'
        line: 'active_on_launch: True'
        backrefs: yes
      become: yes
      become_user: "{{ user.user }}"
      register: mozc_configuration
      # Does this really work?
```

```
    # This might not work, ibus is a user daemon and not a sytem daemon. So
      # this execued as user can do the trick:
      #               - ibus-daemon -drx
      #               - ibus restart
    - name: "{{ ns }}: Restart ibus-daemon for the specific user"
      ansible.builtin.shell: |
        export DISPLAY=:0
        ibus-daemon -drx
      become: yes
      become_user: "{{ user.user }}"
      environment:
        DISPLAY: ":0"
      when: mozc_configuration.changed or dconf_output.changed
  handlers:
    - name: "{{ ns }}: Reconfigure locales"
      listen: "Reconfigure locales"
      ansible.builtin.shell:
        cmd: dpkg-reconfigure --frontend=noninteractive locales
      become: yes
    - name: "{{ ns }}: Configure im"
      listen: "Configure im"
      ansible.builtin.command:
        cmd: im-config -n ibus
      become: yes
      become_user: "{{ user.user }}"
      environment:
        DISPLAY: ":0"
```

4 Debian 10 (buster) [0.1.7]

To understand which tasks are available for buster you can use tasksel.

```
tasksel --list-tasks
tasksel --task-desc task
```

However that do not list anything regarding `Japanese`. The tasksel tool seems rather useless, the following gives nothing.

```
tasksel --task-desc task-japanese
```

We try the Jessie way:

```
aptitude install task-japanese task-japanese-desktop
```

Which will install the following packages:

```
anthy anthy-common fbterm firefox-esr-l10n-ja fonts-ipafont
fonts-ipafont-gothic fonts-ipafont-mincho fonts-vlgothic im-config
kasumi
libanthy1 libanthyinput0 libgcroots0 libm17n-0 libotf0 libprotobuf17
libreoffice-help-ja libreoffice-l10n-ja libuim-custom2 libuim-scm0
libuim8
libx86-1 libzinnia0 lv m17n-db manpages-ja manpages-ja-dev mozc-data
mozc-server mozc-utils-gui nkf psf-unifont task-japanese
task-japanese-desktop tegaki-zinnia-japanese ttf-unifont uim uim-anthy
uim-data uim-fep uim-gtk2.0 uim-gtk2.0-immodule uim-gtk3 uim-gtk3-
immodule
uim-mozc uim-plugins uim-qt5 uim-qt5-immodule uim-xim unifont
xfonts-unifont
```

Read ‘For Debian Jessie’ for more info.

Last step:

```
dpkg-reconfigure locales
```

Leave `en_US.UTF-8` selected and add `ja_JP.UTF-8`

Then log out

Remark: After log in again. A small window with a blue object inside appears in the lower right corner. The window changes shape and size when opening application. It also follows the mouse like a magnet. This is quite annoying. If you quit the toolbar you have to log out and in.

You can access parts of the input system scattered around in the menu system:

```
System->Preference->Other->Anthy Dictionary editor
System->Preference->Other->Input Method
System->Preference->Other->Mozc setup
System->Preference->Personal->Input Method (yes a different one)
System->Control Center->Other->Anthy Dictionary editor
System->Control Center->Other->Input Method
System->Control Center->Other->Mozc setup
System->Control Center->Personal->Input Method
```

Or via

5 For Debian Jessie

Japanese is a very new language with a unique writing and it is around just some many hundred years. Also modern software did not evolve until recently. That might be some factors the input system changes with every release. I hope to the better.

Some stuff has changed in the last years, so Jessie is different.

```
ibus became more popular
Gnome desktop is not so usuable any more, use MATE
new engine from Google
```

So we do:

```
 aptitude install task-japanese task-japanese-desktop
```

This will install many dependencies, including the following:

```
1       ibus ibus-mozc ibus-tegaki im-conig mozc-utils-gui
```

You might consider

```
1  ibus-emoji
2  ibus-latex
```

Last step:

```
dpkg-reconfigure locales
```

Leave en_US.UTF-8 selected and add ja_JP.UTF-8

Then log out

6 For Debian Wheezy

```
1  aptitude install task-japanese task-japanese-gnome-dekstop
```

Then use Anthy and shift+space to activate.

Or (!) if you prefer handwriting:

```
aptitude install task-japanese-gnome-dekstop
ibus
ibus-anthy
ibus-tegaki
```

Then log out of the desktop manager When logged in again, choose input method “anki” in ibus gnome panel-app and choose input method “tegaki” in ibus gnome panel-app as 2nd.

6.1 Hand writing

```
uim-tomoe-gtk?? Missing
uim-dict-gtk3 Missing

gweai,ibus-tegaki,kankipad,tegaki-recognize
```

7 For Debian up to Squeeze

1 locale 2 input conversion 3 ithings NOT to install 4 configuring scim daemon 8 changing anthy 9 further reading

7.1 Step 1 locale

As root:

```
aptitude install locales
```

```
vim /etc/locale.gen
ja_JP.UTF-8 UTF-8
de_DE.UTF-8 UTF-8
en_US.UTF-8 UTF-8
```

```
dpkg-reconfigure locales
# OR
locale-gen
```

7.2 Step 2 input conversion

```
aptitude install scim anthy scim-anthy ttf-kochi-mincho
 ↪  ttf-kochi-gothic
```

on lenny this will install additionally: im-switch{a} libanthy0{a} # libscim8c2a{a} scim-gtk2-immodule{a} scim-modules-socket{a} # ttf-sazanami-mincho{a}

7.3 Step 3 things NOT to install

```
scim-modules-table{a} scim-tables-ja
```

From Package description: Users who need to input Japanese all the time should look at other SCIM modules for Japanese, such as scim-anthy package.

```
scim-modules-socket
```

From Package description: SCIM can use a local or inet socket as the front end and connect to the configuration and IM engine modules. Then other computers and/or environments can share these input methods by connecting to the socket with socket IM engine module and socket configure module.

7.4 Step 4 configuring the scim daemon

```
vim /etc/scim/global
```

```
change
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/SupportedUnicodeLocales = en_US.UTF-8
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

to

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/SupportedUnicodeLocales = en_US.UTF-8,ja_JP.UTF-8,de_DE.UTF-8
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

This need to be in .zshrc:

```
export XMODIFIERS="@im=SCIM"
export GTK_IM_MODULE="scim"
```

If you use GTK_IM_MODULE=“xim”, you will not see anthy in scim

7.5 Step 8 changing anthy

```
aptitude install kasumi
```

7.6 Step 9 further reading

- http://www.h4.dion.ne.jp/~apricots/scim-anthy/howto.html
- http://dspnet.fr.eu.org/~lonewolf/LinuxJapan/Howto_English_Japanese.html

7.7 Step 0 Untested

Better looking non free fonts: ttf-ipa-font

```
1  Edit for screen (.screenrc)
2  defencoding UTF-8
3
4  Edit for Vim(.vimrc)
5  set termencoding=utf-8
6  set fileencodings=iso-2022-jp,cp932,utf-8,japan
7
8  Edit for emacs (.emacs)
9  (when (<= emacs-major-version 21)
10 (require ‘un-define))
11 (set-language-environment “”Japanese)
12 (set-terminal-coding-system ‘utf-8)
13 (set-keyboard-coding-system ‘utf-8)
14 (set-buffer-file-coding-system ‘utf-8)
15 (set-selection-coding-system ‘utf-8)
16 (setq default-buffer-file-coding-system ‘utf-8)
17 (prefer-coding-system ‘utf-8)
18 set ambiwidth=double
```

8 Older Attempts before 2006-04-25

8.1 Japanification under Debian

```
apt-get install scim canna kinput2-canna .......

apt-get install scim scim-uim scim-pinyin scim-hangul scim-chewing
scim-m17n scim-prime scim-anthy scim-skk scim-canna scim-gtk2-immodule
m17n-env ttf-arphic-gbsn00lp ttf-arphic-gkai00mp ttf-arphic-bsmi00lp
scim-tables-additional scim-tables-ja kasumi ttf-unfonts uim-gtk2.0
 ↪  uim-xim

apt-get install canna canna-shion
apt-get install kinput2-canna:

apt-get remove ttf-arphic-gbsn00lp ttf-arphic-gkai00mp ttf-arphic-bsmi00lp

# su - $USER
$ user-ja-conf

# vim /etc/locale.gen

# $USER 2006-03-29
ja_JP.EUC-JP EUC-JP
ja_JP.UTF-8 UTF-8

# locale-gen
```

```
    $ export GTK_IM_MODULE=xim
    $ export XMODIFIERS=@im=SCIM
    $ scim -d
```

9 History

Version	Date	Notes
0.1.8	2024-06-24	Debain 12 bookworm (Initial release quick guide)
0.1.7	2019-08-16	Debian 10 buster
0.1.6	2018-08-10	
0.1.5	2015-10-07	

Version	Date	Notes
0.1.4	2013-07-29	
0.1.3	2013-06-23	
0.1.2	2012-08-27	
0.1.1	2009-07-05	
0.1.0	2006-04-25	Intial release

10 Disclaimer of Warranty

11 Limitation of Liability