いつでもどこでもEmacsライクなキーバインド

伊藤 誠[†]

1 はじめに

パーソナルコンピュータ (パソコン) 等を利用する際に、最も重要となるユーザーインターフェイス (入力装置) はキーボードであろう。マウスやタッチパネル等も補助的に用いることが出来るが、文書を入力する場合などキーボードが本質的な役割を果たしていることに疑問の余地はない。

文書を編集、あるいは他の作業を効率よく行うために、キーボードには様々な機能がショートカットキーとして割り当てられている。従ってパソコンを利用する際、作業効率化のためにショートカットキーを多用するユーザーも大勢いる。

さて、ここで問題になるのが、それぞれのショートカットキーにどのような機能が割り当てられている (バインディング (アサイン) されている) かである。Windows の場合、標準で全てのアプリケーションソフトに対し、ショートカットキーには共通の機能が割り当てられているため、どのアプリケーションソフトを利用しても、共通のショートカットキーで操作することが出来る。

筆者は Windows が普及し始める以前、すなわち Windows のショートカットなど知る由もなかった時代から、コンピュータを利用してきた。また入力用のアプリケーション、すなわちエディタとして UNIX 系の OS(Linux 等) でよく用いられる Emacs を利用してきた。そして現在も利用している。入力・修正を効率よく行うには、当然 Emacs のショートカットキーを多用するわけだが、Emacs のショートカットキーに割り当てられている機能は、Windows のそれとは全く異なっている。つまり、Emacs を昔から利用し、Emacs のショートカットキーに慣れ親しんでいる者にとって、Windows のショートカットキーは異質のものとなる。

その代表的な例が、「Ctrl+p(コントロールキーを押しながら p キーを押す)」であろう。Emacs では、このショートカットキーにはカーソルキーの「上矢印」に相当する機能が割り当てられている。つまり「Ctrl+p」で一行前にカーソルを移動させることが出来る。

しかし Windows では「Ctrl+p」には印刷機能が割り当てられているため、Emacs ユーザがカーソルを一行前に移動させようとして無意識に「Ctrl+p」を押すと、いきなり印刷画面が現れて面食らうことになる。「コントロールキー」+「一文字」という重要なショートカットに、もはや滅多に利用することのない「印刷」という機能が割り当てられていることは、Emacs ユーザにとって耐え難いものがある。

どの様なショートカットキーを利用したいかは個人の好みにもよるが、重要なことは「統一されたショートカットキーでパソコンを操作出来る」ということである。各場面毎に、ショートカットキーにどのような機能が割り当てられていているのかを気にしながら操作しなければならないのは非常に非効率的である。

[†]大阪産業大学 経済学部 経済学科 教授
草 稿 提 出 日 6 月 24 日
最終原稿提出日 6 月 28 日

Linux 等で Emacs を利用しているユーザの多くは、パソコンを利用する際に Emacs のショートカットキーを利用してパソコンを操作したいというこだわりがある。かく言う筆者自身もその一人である。パソコンでの作業が Emacs だけで全てが完結すのであれば問題はないのだが、Emacs ユーザであっても、Emacs 以外の Linux アプリケーションを利用する場合もある。そしてその Linux アプリケーションには、Emacs とは異なるショートカットキーを用いているものが多数ある。また当然のことながら Linux には存在しない Windows のアプリケーションを利用しなければならない場合もある。各オペレーティングシステム毎、あるいは各アプリケーション毎にショートカットキーによる操作方法が異なるのはパソコンを利用する上で好ましいものではない。大袈裟かも知れないが、多大なストレスの原因にも成り得るのである。

筆者は長年 Linux を中心にパソコンを利用してきた。さらにその中心には常に Emacs があった。全ての場面で Emacs ライクにキーバインドされたショートカットキーで、統一的にパソコンを操作したいと常日頃思ってきた。

本研究ノートでは Linux と Windows を、さらに仮想マシンとしての Windows も含めて、全て『Emacs ライクにキーバインドされたショートカットキーで操作する』ことを実現出来たこと。そして、それを実現する際に行ったことを記した備忘録である。

2　Linux(Ubuntu) と Windows

2.1　Linux(Ubuntu) の現状

「はじめに」にも述べた通り、筆者は長年 Unix ライクな OS である Linxu を愛用して来た。またここ 10 年以上は Linux のディストリビューションの一つである Ubuntu を利用している。Linux では GNU が提供する多くのフリーソフトウェアを利用することができる。その中の一つとしてエディタの Emacs があるわけだが、Emacs を利用すると『カーソル移動にカーソルキーを必要としない』ため、Emacs ユーザは様々な場面でカーソルを移動させようとして、Emacs のカーソル移動のためのショートカットキー (「Ctrl+p」など) を無意識のうちに使ってしまう。

しかし実際のところ、Linux は GNU のソフトウェアにかなり依存しているにもかかわらず、Windows ユーザを取り込むためなのか、多くの場面で Windows ライクなショートカットキーが用いられている。そのため「はじめに」にも書いた通り、Linux を利用していたとしても、「Ctrl+p」を押すと印刷画面が表示されることが多々ある。

Linux のアプリケーションには Emacs ライクなショートカットキーの割当を選択できるようになっているものも多くある。しかし全てではない。つまりアプリケーション毎にショートカットキーが異なることになる。Linux はショートカットキーによる操作に関しては、全くと言っていい程統一性がないのが現状である。

残念なことに、これを解決する方法を、筆者はなかなか見つけることができなかった。

2.2　Windows と keyhac

翻るに、Windows は全ての場面で、統一されたショートカットキーを利用することが出来る。ならば Emacs ライクにこだわることなく、Windows のショートカットキーに慣れれ

ばよいのかも知れないが、何故か Windows には『カーソル移動のためのショートカットキー』がない。カーソル移動という重要な操作を、Windows ではカーソルキーを用いて行なわなければならない。Emacs ユーザにとって、これも耐え難いことの一つなのである。つまり Emacs ユーザはカーソルを移動させるために、一々キーボードのホームポジションから手を離すことはしたくないのだ[1]。

ところが、Windows にはショートカットキーをカスタマイズすることを可能にする keyhac[1] というソフトウェアがある。これはプログラミング言語 Python を用いて記述されていて、fakeymacs[2] なるものを併せて用いると、Windows のキー操作を容易に Emacs ライクにカスタマイズすることが出来る。つまり keyha+fakeymacs を用いると、Linux の中途半端な Emacs らしさよりも、なんと Windows の方がより Emacs らしくなるわけである。このため筆者は自らのパソコン環境として、本来ならば Linux(Ubuntu) を中心に利用したいにもかかわらず、keyhac の存在を知り、自身の Windows マシンに keyhac と fakeymacs を導入して以来、パソコンを操作するインターフェースとして Windows マシンを利用する機会が多くなった。

keyhac にはキーボードを Emacs ライクにする以外にも、様々な機能があるが、取り敢えず筆者の「Emacs ライクなキーバインドにする」という目的は、keyhac と fakeymacs によって Windows 環境では達成することが出来たのである。

導入の手順を簡単にまとめておく。まず keyhac をインストールする[2]。次に keyhac のアイコンを右クリックして表示される「設定の編集 (E)」をクリックするとメモ帳が起動するので、その内容を fakeymacs に含まれる `config.py` の内容とそっくり入れ替えて保存する。そして「設定のリロード」をクリックすればよい (図 1 参照)。あるいはこの `config.py` を keyhac の実行ファイルと同じフォルダに保存して keyhac を起動すればよい。`config.py` は Python で記述されているので、その内容を修正することにより自分独自のキーバインドにカスタマイズすることも可能である。また、`config.py` の修正は、必要なときにだけ行えばよい。常に keyhac を使用するのであれば、keyhac を自動起動するように設定しておく。

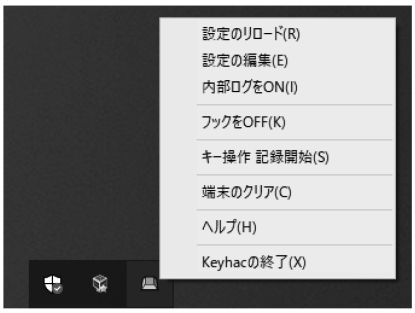

図 1: keyhac のコンテキストメニュー

2.3　Linux(Ubuntu) と xkeysnail

Linux では Emacs とターミナルを使う場合が多く、ターミナルも Emacs ライクなショートカットキーを利用することが出来るので、Windows をインターフェースとして Linux を利用しても余り不便を感じることはない。しかし、Linux ユーザにとって「Windows は単なるユーザーインターフェイスである」と思い込もうとしても、なかなか割り切ることは出

[1] このことが、Emacs ユーザに Emacs ライクにバインドされたショートカットキーを使いたいと思わせる、最大の理由かも知れない。

[2] 必要であれば Visual Studio 2015 ランタイムパッケージもインストールする。

来ない。あくまでも「パソコンは Linux を中心に据えて使用し、必要があれば Windows も利用する」という使い分けが、個人的には望ましい利用方法である。そこで根本的に Linux 全体が Emacs ライクにならないものかと実に長い間思い悩んで来た。ところがつい最近であるが、xkeysnail[3] なるものがあることを知った。xkeysnail を用いると、Linux(Ubuntu) を Emacs ライクなショートカットで操作可能となる。

xkeysnail がどのようなものであるかの詳細は、作者のページ [4] を見て欲しい。xkeysnail の特徴としては、Python で記述されていて、起動には root 権限が必要なこと等が挙げられる。

実際に自身の Linux(Ubuntu) 環境に xkeysnail を導入したのだが、導入にあたっては「xkeysnail でキーリマップする」[5] を参考にした。

xkeysnail の導入方法を以下に簡単にまとめておく。

1. xkeysnail のインストール

   ```
   sudo pip install xkeysnail
   ```

2. uinput グループの追加

   ```
   sudo groupadd uinput
   ```

3. xkeysnail ユーザの作成

   ```
   sudo useradd -G input,uinput -s /sbin/nologin xkeysnail
   ```

4. xkeysnail ユーザに/dev/input 及び/dev/uinput へのアクセス権を与えるため、以下の 2 つのファイルを以下の内容で作成

   ```
   /etc/udev/rules.d/40-udev-xkeysnail.rules
   KERNEL=="uinput", GROUP="uinput"

   /etc/modules-load.d/uinput.conf
   uinput
   ```

5. userA が xkeysnail に sudo して xkeysnail をパスワード無しで実行出来るように、以下のファイルを以下の内容で作成

   ```
   /etc/sudoers.d/10-installer
   userA ALL=(ALL) ALL,\
            (xkeysnail) NOPASSWD: /PATH_to_xkeysnail/xkeysnail
   ```

   userA と PATH_to_xkeysnail は、各自の環境に合わせて変更する。

6. 自動起動スクリプトの例

   ```
   #!/bin/sh
   if [ -x /PATH_to_xkeysnail/xkeysnail ]; then
     xhost +SI:localuser:xkeysnail
   ```

```
    sudo -u xkeysnail  DISPLAY=$DISPLAY /PATH_to_xkeysnail/xkeysnail \
                       /PATH_to_config_file/config.py &
fi
```

PATH_to_config_file も各自の環境に合わせて変更する。このスクリプトを起動時に実行するよう設定しておく。

上記「6.」の中にある config.py がキーバインドを記述したファイルである。このファイルは前述 [3] の example ディレクトリ内にある。これをそのまま使用すれば、Linux のショートカットキーがほぼ完璧に Emacs ライクになる。

なお config.py の中に

```
# Emacs-like keybindings in non-Emacs applications
define_keymap(lambda wm_class: wm_class not in ("Emacs", "URxvt"), ...
```

という記述がある。ここの右側のカッコ内に、「キーバインド変更の対象外」とするアプリケーション名 (WM_CLASS) を列挙する。この例では Emacs とターミナルの URxvt を対象外としている。もともとキーバインドが Emacs ライクであるアプリケーションを除外するためである。他にも除外したいアプリケーションがあれば、この部分にその名前を書いておけばよい。そのためにはアプリケーションの WM_CLASS を調べなければならない。以下のコマンド

```
xprop WM_CLASS
```

を入力してから、マウスで当該ウィンドウをクリックすると、例えば gnome-terminal の場合

```
WM_CLASS(STRING) = "gnome-terminal-server", "Gnome-terminal"
```

と表示される。最後に書いてある名前が gnome-terminal の WM_CLASS 名である。

xkeysnail ユーザを作成する理由など、設定の詳細については前述の「xkeysnail でキーリマップする」[5] を参照して欲しい。また config.sys ファイルの記述方法は、作者のページ [4] を見て欲しい。

3　ショートカットキーの統一

以上で、Linux と Windows のいずれも、Emacs ライクなショートカットキーで、統一した操作が出来るようになった。

「ショートカットキーの統一」を行い、「いつでもどこでも Emacs ライクなキーバインド」を実現するのには意外と長い時間がかかってしまった。しかし何はともあれ長年の望みだったことが実現したことは、些細なことかも知れないが、個人的には望外の喜びである。

ところで、じっと待つくらいなら、いっそ自分でプログラムすれば良いのに、と思われる諸兄もいるかも知れない。しかし数値計算を主体にコンピュータを利用してきた筆者にとって、システム周りのプログラミングはかなり敷居が高いものがある。

逆に、じっと待ち続け、諦めずに探し続けていたからこそ実現できた、と言えるかも知れない。

4　Linux(Ubuntu) と Windows 環境のさらなる統一

Linux ユーザで、全てを Linux で完結させている人は意外と少ないと思われる。私自身 Windows を利用して授業を行うため、Windows の利用は避けて通ることができない。また有用なアプリケーションソフトで、Windows でしか利用できないものもある。Mac に関しては個人的には無理して使わなくとも済むので、普段利用することはない。

つまり日常的には、Linux と Windows の両方を同時に利用し、使い分けることになる。すると当然 Linux 用と Windows 用の 2 台のパソコンを常時動かさなければならない。両者を使い分ける方法はいろいろある。例えばパソコン 2 台とディスプレイ 2 台を用意して、synergy[6] 等を利用して一組のキーボードとマウスを共有して両者を操作する方法。あるいは、PC 切替器を使ってディスプレイとキーボード・マウスを 1 組用意し、切替器で 2 台のパソコンを切り替えながら利用する方法等が考えられる。

どちらも一長一短があるが、いささか冗長 (少なくとも 2 台のパソコンを動かさなければならない[3]) であることは否めない。1 台のパソコンと 1 つのディスプレイ、それと一組のキーボード・マウスを使って、同一のディスプレイ上で Linux と Windows を使い分けられたら格段に自由度が増し、効率的である。これを実現する方法として、いわゆる仮想マシンがある。

実は筆者は 10 年以上前から、仮想マシンを利用したり、止めたりを繰り返してきた。しかし実際に仮想マシンを使ってみると、その性能・操作性が、単体で利用する場合に比べ著しく劣ってしまうため、久しく仮想マシンは利用していなかった。しかし近年、コンシューマ向けの CPU が 6、8、10… コアと顕著な性能向上が見られ、仮想マシンもかなり快適に動くようになった。Linux が Emacs ライクになったので、Linux を中心に利用し、VirtualBox を使って Windows を仮想マシン (ゲスト OS) として利用することにした (図 2)。

ところが、仮想マシンの Windows に keyhac をインストールすると、Windows 側でショートカットキーがうまく動作しない。理由は Linux(ホスト OS) 側から見ると、仮想マシン (ゲスト OS) は単なる X ウィンドウのアプリケーションソフトに過ぎず、xkeysnail でカスタマイズした Linux 側のキーバインドがそのまま仮想マシンにも適用されてしまうため、keyhac と両立しないのである。

図 2: ゲスト OS としての Windows

つまり、逆に仮想マシンの Windows では keyhac を動かしてはいけない。動かさなければ Linux 側のショートカットキーがそのまま仮想マシンの Windows でも機能する。これで全て丸く収まった。

ここで Emacs 自体について一言述べておこう。Emacs はエディタアプリと言うよりも、実際のところ elisp(Emacs Lisp と呼ばれる Lisp の一種) のインタープリタである。素の Emacs(デフォルトの Emacs) は核となるインタプリタに、必要最小限の (と言っても膨大

[3] 1 台のパソコンでデュアルブートする方法もあるが、これでは同時に両者を利用することは出来ない。切替器でも両者を同時に利用することは出来ないが、切替に要する時間は無視出来る。

な) elisp を実装したエディタであると思えばよい。これらの elisp に加えて、さらに Emacs の初期化ファイルを自ら elisp を用いてカスタマイズすることにより、Emacs は世界で唯一つの、そして極上の「マイ」エディタとすることが出来る。しかし残念なことにカスタマイズし過ぎると、カスタマイズの経緯を知っているのは自分だけになってしまう。多くのカスタマイズを施すと、自身が行ったカスタマイズにもかかわらず、時間とともに何をしたのか忘れてしまう。また Emacs 本体のバージョンが変わると、今まで動作していた elisp が突如として動かなくなることがある。つまり「マイ」エディタを管理できるのは世界中で自分一人だけなのである。以前「プログラム実行環境としての Emacs」[7] という研究ノートを本論集に寄稿したことがあるが、実際のところその内容はカスタマイズ直後は理解していても、紆余曲折 (elisp が動かなくなったり、カスタマイズした機能を久しく使わなくなったり) を経て、現在はほぼ完全に忘却の彼方にあるのが実情である。実際 Emacs を保守し続けることは相当の気力と、手間と、労力を要する。

筆者は現在でも Emacs を利用しているが、このような理由で、他に Emacs に代わるエディタで Linux でも Windows でも動作するものがないか探していた。するとその代表的なものに、Visual Studio Code(VSCode) や Atom 等があることが分かった。

VSCode では拡張機能、また Atom はパッケージを付け加えることによってカスタマイズすることが出来る。各拡張機能・パッケージを利用しているユーザーは世界中に沢山いるので、問題が生じてもネット等で調べて解決することが出来る。エディタ本体及び拡張機能・パッケージのバージョンも自動的に更新してくれる。またいろいろなパソコンに VSCode あるいは Atom をインストールしたとしても、相互のパソコン間で設定の同期を自動で行うことが可能である。これら諸々の理由で VSCode、あるいは Atom を使うのも良いと思うようになった。

しかし、VSCode に関しては、何故か xkeysnail とも keyhac とも相性が良くない。VSCode が Emacs ライクにならないのだ。そのため VSCode の拡張機能である「Awesome Emacs Keymap」を利用してみることにした。この拡張機能は名前の通り、VSCode のキーバインド (キーマップ) を Emacs ライクにするものである。これによって VSCode も Emacs ライクに利用することが出来るようになる。

VSCode そのものが Emacs ライクになったので、Linux 側では xkeysnail の対象外リストに VSCode を書き加えれば Linux 版 VSCode では問題は生じなくなる。しかし今度は、仮想マシン (Windows) 側で Windows 版 VSCode を使うと、Linux 側のキーバインドと、「Awesome Emacs Keymap」が競合してしまい、うまく動作しない。

何とも嘆かわしい限りであるが、解決策は意外と簡単であった。先程も述べた通り Linux 側から見た仮想マシンは、あくまでも単なる X ウィンドウアプリケーションである。つまりこの仮想マシンである VirtualBox そのものを、xkeysnail の対象外とし、仮想マシンの Windows 側で keyhac を動かすのである。すると、仮想マシン内では xkeysnail ではなく、keyhac がキーバインドを制御することとなり、VSCode で「Awesome Emacs Keymap」を利用すれば Windows との相性も、単独で動かした場合と同様になる。

VSCode を利用する場合、最終的に `config.py` 内の、xkeysnail の対象外とするアプリケーションソフトを列挙した部分は

```
# Emacs-like keybindings in non-Emacs applications
define_keymap(lambda wm_class: wm_class not in ("Emacs", "URxvt",
```

```
"Terminator", "Gnome-terminal", "Code", "VirtualBox Machine"), ...
```

となる。

では次に Atom エディタはどうであろう。Atomo は VSCode と異なり xkeysnail との親和性がかなり高い。キーバインドを Emacs ライクにするためのパッケージは存在するが、それを使用する必要ない。すると仮想マシンを xkeysnail の例外リストに列挙する必要がなくなる。つまり Atom と VSCode は並立しないことになる。

VSCode と Atom はエディタとしての機能には余り大きな差はないようであるが、本稿執筆段階で判明している問題点として、VSCode は拡張機能で Emacs ライクになるのだが、細かい点で Emacs ライクなキー操作が出来ない部分が多数ある。また、Atom は xkeysnail との親和性は高いが、やはりいくつかの点で Emacs ライクにならない。また Web ベースのエディタのため、日本語入力時の文節の切れ目が全く分からない。

両者とも問題点はあるようだ。どちらのエディタを Emacs に代わるエディタとして利用するかは好みによると思われるが、筆者はまだ両者を利用し始めて日が浅いため、いずれが良いかはまだ判断出来ないのが現状である。しかし、どちらにも問題点があり、今後さらなる問題点も顕在化してくるかも知れない。これらの問題を解決できるかどうかも不明である。Emacs の代替エディタの模索はこれからも続きそうである。もしかしたら結局本家の Emacs をこれからも使い続けることになるかも知れない。

Emacs の代替エディタ (不完全で多少問題は残しつつも) を含め、以上でホスト OS としての Linux と、仮想マシンのゲスト OS である Windows が 1 台のパソコンで動作し、ともに Emacs ライクなキーバインドで統一して操作することが可能となった。

5　まとめ

私自身 Emacs を使い始めて 30 年以上立つ。この間、パソコンをいつでもどこでも Emacs ライクなキーバインドで使えたら良いのにと思い続けてきた。この思いがようやく 30 年の時を経て実現することとなった。

一時期、Windows と keyhac で Emacs ライクなキーバインドを施した Windows パソコンが、筆者が利用するパソコンの中心なったことがあった。しかし xkeysnail を用いることにより、以前のように Linux 中心のパソコンライフに戻ることが出来た。

そもそも基本的に Windows は何をするにも ”動作が遅い ” と言う印象がある。Linux や Mac はその点、キビキビと動作する。この研究ノートも、Linux 上の LaTeX システムを利用している。Windows でも LaTeX を利用することは出来るのだが、Linux でのコンパイルのスピードは Windows の比ではない。Windows の LaTeX の処理が遅過ぎる。Emacs も Windows で動くのだが、起動は遅いし、何より Emacs の初期化ファイルを Linux と完全には共有することができない。

さらに Windows はライセンス認証の問題もある。無償でどこでも利用できる Linux との差は歴然である。例えば図 3 のようなリムーバブルケース[4] を用いると、Linux をインストールしたディスクあるいは SSD 等を自宅のパソコンから取り外し、職場へ持っていって、職場のパソコンに取り付ければ、そのまま職場でも Linux を使うことが出来る。つ

[4] https://www.ratocsystems.com/products/subpage/sa25rc1x.html

まりディスクあるいは SSD 一つで、自宅でも職場でも、あるいは職場以外の任意の場所で、全く同一の環境で作業をすることが出来る。また、たとえパソコンの設定等に修正を加があったとしても、その作業は 1 回だけで済むことになる。この利便性は代えがたい。

これらが Linux を中心にしてパソコンを利用したいと思う主な理由である。

もっとも、オンラインストレージを利用すれば、データの共有は可能である。アプリケーションソフトやパソコンの設定を完全に同一にすることは、労を惜しまなければ Windows でも可能かも知れない。しかし高価な有償ソフトやライセンスが絡むソフトを、全ての Windows パソコンで利用するのは難しい。ライセンス等を気にすることなく、Linux を中心に据えてパソコンを利用することが、筆者にとっては健全なパソコンの利用方法なのである。デスクトップパソコンではなく、ノートパソコンを利用する方法もあるが、持ち運ぶことを考えると、ディスク (SSD) 一つと、軽くなったとはいえノートパソコンとの重量差は比較にならない。またデスクトップパソコとノートパソコンの性能差は歴然である。

図 3: ラトックシステムのリムーバブルケース

しかし Linux を中心に据えたとしたとしても、前にも述べたが、Windwos を利用することなく全てが Linux で完結するわけではない。MS Office 関連の資産が余りに莫大なため、なかなか Windows なしというわけには行かない。また Windows でしか利用できない有用なアプリケーションソフトは実際山ほどある。

すると次のような使い方も出来ることに思い当たる。Windows を仮想マシンとして Linux の中に住まわせてあげるのである。すると Linux を持ち運ぶことにより Windows 環境も、Windows のライセンスも Linux と一緒に移動することになる。このようにして仮想マシンを利用すれば、Linux はそもそもライセンス認証を気にする必要がないので、ポータブルな Linux ＋ Windows 環境を容易に得ることができる。先にも述べたが、仮想マシンと言えども近年は CPU の性能向上で、独立した単体として使用する場合と比べて、ゲームでもしない限り実用的な性能で利用することが出来る。従って Linux+仮想マシン (Windows) という選択肢は、今や現実的なものとなった[5]。

これで Linux ユーザは、1 台のパソコンを「Linux+仮想マシン (Windows)」とすることにより、2 つのオペレーティングシステムを、統一されたショートカットキーで、シームレスに使用することが可能となる環境を手にすることが可能となった。これで、『いつでも (どのアプリケーションを利用する時でも)、どこでも (Linux であっても Windows であっても)、Emacs ライクなキーバインド』でパソコンを操ることが出来るわけである。

最後に日本語入力について付け加えておくと、Google 日本語入力 (mozc) は、Linux でも Windows でも利用可能である。また日本語入力のキー操作を Emacs でよく用いられる

[5] 自宅と職場等に仮想マシンを快適に動作させるに足るパソコンを用意しなければならない、という問題は残るかも知れない。

egg[6]ライクなキーバインドにカスタマイズすることも出来る。このように日本語入力も Linux 及び Windows 共に、Emacs ライクなショートカットキーで統一して操作可能となる。

以上が、『筆者が 30 年以上追い求めてきた、いつでもどこでも Emacs ライクなキーバインドで統一的にパソコンを操作することが可能となった』顛末記である。

URL・参考文献

[1] https://sites.google.com/site/craftware/keyhac-ja

[2] https://github.com/smzht/fakeymacs

[3] https://github.com/mooz/xkeysnail

[4] https://qiita.com/mooz@github/items/c5f25f27847333dd0b37

[5] https://qiita.com/miy4/items/dd0e2aec388138f803c5

[6] https://symless.com/synergy

[7] 伊藤 誠 (2013) プログラム実行環境としての Emacs(大阪産業大学経済論集 Vol. 14 No. 2)

[6]昔の Emacs の代表的な FEP。ちなみに egg は、「たくさん、またせて、ごめんなさい」の頭文字「たまご」→「egg」(英訳) に由来する。恐らく egg の作者は、開発にかなりの時間を要したのであろう。

Emacs-Like Key Bindings Anytime, Anywhere

ITOH Makoto

Key Words: Emacs, Key Bindings, Linux, Windows, Virtual Machine

Abstract

The keyboard is an essential device when using a computer. To perform editing tasks efficiently, it is important to know what functions are bound to the shortcut keys. Those familiar with Emacs-like key bindings would, most likely, find operating a computer eve easier with the use of Emacs-like key bindings. Using xkeysnail on Linux and keyhac on Windows makes it possible to operate a computer with Emacs-like key bindings on both Linux and Windows. I also demonstrate how to achieve Emacs-like key bindings on a virtual machine under Linux.