Shell and Emacs with little customizations

田浦

すべての目的

- 作業のピークスピードの向上
- 知っとけば作業効率がぐんとあがること
- だが，意外と使われていない (知られていない) と感じること
  - 入力の削減
  - マウスを使わない，遠いキー (e.g., 矢印キー) を使わない，長距離を一気に移動する，etc.
- 注: ここでは,「カスタマイズしまくってめっちゃ便利に」みたいなことをして，人を引かせるようなことはしません

以降すべての前提

- まずは ctrl キーを「正しい」位置に直す!

```
1 $ gnome-tweak-tool
```

タイピング →Ctrl キーの位置

始める前に...

日本国憲法第 20 条

- 信教の自由は、何人に対してもこれを保障する。いかなる宗教団体も、国から特権を受け、又は政治上の権力を行使してはならない。
- 何人も、宗教上の行為、祝典、儀式又は行事に参加することを強制されない。
- 国及びその機関は、宗教教育その他いかなる宗教的活動もしてはならない。

日本国憲法第 21 条

集会、結社及び言論、出版その他一切の表現の自由は、これを保障する。*(以下略)*

補完 (シェル, Emacs 共通)

2,3 文字に一回は **タブ** を打とう

- タイプ時間の節約
- タイプミスの削減

シェル: ヒストリ

- 上矢印 (最近入れたコマンドを引き出す)
- C-r ... (... に入れた文字列を含むコマンドを引き出す)
- どちらも，引き出したあとで編集可能

```
1 CFLAGS="-O0 -g" ./configure --prefix=$HOME/install/gnuplot
```

なんて打つのは一日一回でたくさん!

- シェルのコマンドライン編集のキー操作は，Emacs と共通 (C-a, C-e など)

知らなきゃダメよ〜，ダメダメ，なコマンド

- lv：スクロールしながら長いファイルを表示; / で検索; .gz などは勝手にほどいてくれる
- grep：検索; -r でディレクトリを再帰検索; -I はバイナリファイルを無視; e.g.,

```
1 grep -rI typedef .
```

- find：種々の条件でファイルを検索; e.g.,

```
1 find . -name malloc.c
```

- locate：システム全体でファイル名を探す; e.g.,

```
1 locate *libgtk*.so
```

これから「Emacs」の話をしよう

いまを生き延びるための
エディタ

Emacs: カーソル移動の小技

少し覚えれば確実に少し便利になるいくつかのキー

- C-a (行頭) と C-e (行末)
- バッファの終わり (M->) (同じことだが C-[をおしてから>)

なるべく，矢印キーを押して「つつつつつつ...」をやらないことを心がける

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから
- C-[を押してから

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから
- C-[を押してから ← 実はこれがオススメ

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから
- C-[を押してから ← 実はこれがオススメ

理由:

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから
- C-[を押してから ← 実はこれがオススメ

理由:

- Alt は手を動かす必要が生じる; 「押しながら」は辛いときがある (e.g., Alt + シフト + $\cdots$). キーボードによってはどれがそれなのかわからないことも

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから
- C-[を押してから ← 実はこれがオススメ

理由:

- Alt は手を動かす必要が生じる; 「押しながら」は辛いときがある (e.g., Alt + シフト + $\cdots$)．キーボードによってはどれがそれなのかわからないことも
- ESC は遠い．x とも遠い (M-x)

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから
- C-[を押してから ← 実はこれがオススメ

理由:

- Alt は手を動かす必要が生じる; 「押しながら」は辛いときがある (e.g., Alt + シフト + $\cdots$)．キーボードによってはどれがそれなのかわからないことも
- ESC は遠い．x とも遠い (M-x)
- そいつらは他の目的に奪われたりしがち

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから
- C-[を押してから ← 実はこれがオススメ

理由:

- Alt は手を動かす必要が生じる; 「押しながら」は辛いときがある (e.g., Alt + シフト + $\cdots$)．キーボードによってはどれがそれなのかわからないことも
- ESC は遠い．x とも遠い (M-x)
- そいつらは他の目的に奪われたりしがち
- C-[は安定している

小脱線: メタキーの色々なうち方

M- と言われたら，以下のどれでも良い

- Alt を押しながら
- ESC を押してから
- **C-[を押してから** ← 実はこれがオススメ

理由:

- Alt は手を動かす必要が生じる; 「押しながら」は辛いときがある (e.g., Alt + シフト + $\cdots$)．キーボードによってはどれがそれなのかわからないことも
- ESC は遠い．x とも遠い (M-x)
- そいつらは他の目的に奪われたりしがち
- C-[は安定している
- もちろん Ctrl キーは正しい位置にあることが前提

Emacs: C-s と C-r

- 「C-s 文字列」 で文字列前方インクリメンタル検索
- 「C-r 文字列」 で文字列後方インクリメンタル検索
- 同じ文字列で次を検索したければ，C-s (または C-r) を連打

Emacs: C-s と C-r

- 「C-s 文字列」 で文字列前方インクリメンタル検索
- 「C-r 文字列」 で文字列後方インクリメンタル検索
- 同じ文字列で次を検索したければ，C-s (または C-r) を連打
- もちろん Ctrl キーは正しい位置にあることが前提

Emacs: C-s は他のエディタの検索とは違うのだよ!

- 一瞬で検索を初められ,
- 一文字打つごとにそこまでの文字が検索され,
- 見つかったところですぐにやめればいい.

Emacs: C-s は他のエディタの検索とは違うのだよ!

- (英語の文章やプログラムならば) ほとんど「カーソル移動」の手段と言ってもいいくらい
- カーソル移動は以下で:
  - C-a (行頭), C-e (行末)
  - C-s (前方), C-r (後方)
  - C-f (右), C-b (左), C-p (上), C-n (下)

Emacs: マウスなしのコピペを使いこなそう

- 選びたい領域の,
  - 端っこで C-SPACE (もしかすると C-@; “Mark set”)
  - もうひとつの端っこへカーソル移動
- その状態で
  - cut: C-w
  - paste: C-y
- ちなみに,
  - copy: M-w (つまり C-[w)
  - 別途覚える必要はあまりないという説もある M-w ≈ C-w ; C-y
  - 用途: 編集不可のファイルか
  - 用途: 一瞬でも消すのがためらわれる大きな領域

Emacs: マウスなしのコピペを使いこなそう

C-k が 1〜数行の cut-paste に有用

- 今いる場所からその行末 (の直前) までを cut : C-k
- 直後にもう一度 C-k で行末も cut
- さらに繰り返せば 2 行 3 行... とまとめて cut
- C-a (行頭へジャンプ) と組み合わせて使いこなそう
- 使用例:
  - 1 行まるごと cut

```
C-a C-k C-k
```

  - 1 行まるごと複製

```
C-a C-k C-k C-y C-y
```

  - 3 行まるごと複製

```
C-a C-k×6 C-y C-y
```

コピペ応用編

関数をいっこ，まるごと複製する (いじる前によくやるやつ)

```
1  generic *
2  gp_alloc(size_t size, const char *message)
3  {
4      char *p;^^I^^I^^I/* the new allocation */
5
6  >         ...
7
8      return (p);
9  }
```

- 関数先頭へ移動 (検索で! `C-r generic`)
- マークを残す (`C-SPACE`)
- 関数終りへ移動 (検索で! `C-s ...`)
- cut;paste;paste; (`C-w C-y C-y`)

Emacs: M-x shell のススメ

- Emacs の中でシェルを立ち上げる
- 何が便利? 色々
  - やったことを保存してくれる (configure で何か怒られてなかった???)
  - 結果をコピペ (日記に書く)
  - 前のコマンドの再入力 (コマンド行に戻って ENTER で OK)
  - シェルとの間で行き来の手間が減る
- その他，Emacs の中で生活することのすすめ (M-x grep, gud-gdb, compile, etc.)
- ファイルを編集するたびに立ち上げなおすものではない

Emacs: C-r と M-x shell

- 例 1: Emacs のシェル内で過去に入力したコマンドを探して再入力 ⇒

```
1 C-r プロンプトの文字列 C-r C-r C-r ...
```

- 例 2: configure で何か warning なかった?? ⇒

```
1 C-r warning:
```

- 例 2: 最後の configure で何か warning なかった?? ⇒
  - C-r で一旦最後のコマンドの入力行へ戻り
  - C-s warning で前方に検索

置換

- M-x query-replace (一個ずつ確認しながら置換)
- M-x replace-string (一気に置換)
- キーボードマクロ (これまでに学んだわざと組み合わせて，汎用，強力，かつお手軽な自動化手法)

キーボードマクロ

- `C-x (`：記録開始
- `C-x )`：記録終了
- `C-x e`：記録内容実行
- 回数のバリエーション
  - `C-u C-x e`：記録内容を 4 回実行
  - `C-u C-u C-x e`：記録内容を 16 $(= 4^2)$ 回実行
  - `C-u C-u C-u C-x e`：記録内容を 64 $(= 4^3)$ 回実行
  - `C-u` n `C-x e`：記録内容を n 回実行

キーボードマクロを使う上での要点

- ある意味を持った一連の編集作業が,「全く同じキー操作」でできるようにする
- そのために，これまでの技 `C-a`，`C-e`，`C-s` などが有効
- 「一文字移動」だけでは単語の長さの違いなどを吸収できない

Emacs: 知っとかないといらつく編

- ここまでは,「オフェンス」
- 以下は「ディフェンス」
  - とりたてて「便利」というわけではなく，知っとかないと Emacs 嫌いになる項目
- ひとつ目: コマンドの最中にやめたくなったら `C-g`
- ふたつ目: Undo は `C-_`

Emacs: 窓を整理する

- コマンド (M-x shell, gdb など) を使っていると，勝手に窓が割れる
- それを自在に整理できないとストレス貯まる
  - `C-x 0`：その窓を非表示にする
  - `C-x 1`：その窓だけを表示する
  - `C-x 2`：水平に割る
  - `C-x 3`：垂直に割る
- `C-x o`：窓間の移動
- つまり
  - 勝手に割られた窓がうっとおしければ `C-x 1`
  - また表示したくなったら `C-x b` (割りたければ割る)

Emacs: ファイル間の行き来の仕方

- `C-x b` (switch to buffer)
- ファイル名を覚えていればそれを入力
- ここでもタブ補完!
- いきなりタブで一覧表示
- シェルのバッファは，*shell*

もうひとつ,

- `C-x C-b` バッファ(開いているファイル) 一覧
- カーソルで選んで `f` でそのバッファへ移動

最後に

- ここまで Emacs を一切カスタマイズしなくても出来る.
- 参考までに，私がやっている，重要なカスタマイズはひとつ (だけ)
  - 他のウィンドウに移動するときの `C-x o` を，`C-o` だけに
  - やり方: 以下を ~/.emacs に記入

```
1 (define-key global-map "\C-o" 'other-window)
```