スパルタンVim 5.0

MURAOKA Taro (KoRoN, @kaoriya)

Version 14th Aug, 2016

Table of Contents

はじめに

Vimでは実に多くの機能がキーに割り当てられています。ノーマル、挿入、コマンドライン、3つのモードで大文字小文字区別無しのアルファベットに限定するだけでも、ざっと 3 * 26 で 78 個ものキーにマッピングされた機能が存在していることになります。

一方でVimには自由にキーマップを変更してしまう機能があります。これは個人の好みで自由に操作方法を構築できることを意味しています。しかし無節操なマッピングは、デフォルトのマッピングとの整合性を壊したり、その他のVimの機能との直交性を乱したりする可能性があります。たとえば挿入モードでの迂闊なマッピングは、ドットリピートを妨げることが多々あります。

本書スパルタンVim 5.0では、 デフォルト状態での機能と基本的なキー割り当ての一部について、その背景や利用シーンを解説することで、賢明な読者諸氏により良いキーマップの定義方法について再検討を促してもらおうとするものです。

ノーマルモード

ノーマルモードはVimの基本的な操作モードです。とりわけ編集作業という意味では、 もっとも滞在時間の長いモードになるでしょう。 そのためというわけでもありませんが、 `CTRL` キーなどの修飾キーとの組み合わせもあり、特に多くの機能がキーに割り当てられています。本章ではそれらのキーのうち、修飾の無い基本キーに割り当てられた機能のみを紹介します。

a, A, i, I, o, O

これらのキーはいずれも入力モードへ移行します。 詳細は Vim のヘルプ `:help inserting` を参照してください。

いずれもVimを使う上では欠かせない操作であり、これらをキーマップで潰してしまうのはもってのほか、と言ってよいでしょう。

b, B, e, E, w, W

これらは単語単位の横方向移動です。 ヘルプは `:help word-motions` を参照してください。

特筆すべき事として、小文字はオプション `'isk'` で解釈できる単語に対する操作で、大文字は空白で区切られた単語に対する操作であると覚えおくと良いでしょう。

いずれも頻繁に行なう操作であり、キーマップで潰す余地はほとんどありません。しかし前述の単語の定義では日本語に対してまったく機能しないため、これらを異なる単語の定義に対応する目的でキーマップすることは悪くないアイデアです。

c, C, r, R, s, S

これらは既存のテキストを削除して入力モードに移行します。結果、置き換え操作となります。ヘルプは `:help replacing` に記載されています。

ノーマルモードだけではなく、ビジュアル選択モードからも利用できる直交性があります。また単に置き換えるだけではなく、削除したテキストがレジスタに記録されるため、それを一手先の作業に組み入れた柔軟な使い方のできる機能群です。よってキーマップは避けるのが賢明でしょう。

特に `gR` は、テーブルや日本語編集において絶大な効果を発揮する場合があるので、知らない人は是非一度、試してみることを推奨します。

d, D, x, X

説明するまでもなく必須です。キーマップするのは避けましょう。ヘルプは `:help deleting` です。

f, F, h, l, t, T

左右方向の移動に用います。ヘルプは `:help left-right-motions` を参照してください。アルファベット以外では `;` と `,` の両キーもこのカテゴリーに含まれます。

これらは文句なく利用する機能ですからキーマップは不可です。しかし `f` や `t` に関しては、日本語などの非アルファベットへの対応目的で、適切なキーマップを行なう余地があります。真に適切な方法が未だに存在していないことを除けば、ですが。

g

`g` で始まる機能は非常にたくさんあります。詳細は `:help g` を見てください。 基本的に既存機能の拡張版を `g` プレフィックスにしているようですが、アルファベットでみても思いのほか空きがあります。

これらの空きをキーマップに用いるのは悪くないアイデアです。ただしVim本体の機能が増えた場合には、これらのキーが割り当てられることがよくあります。よって `g` で始まるのキーをマッピングする場合には、将来のVim本体の機能追加に注意が必要となります。

H, L, M

カーソルのある行を画面の上、下、真ん中にスクロールさせる機能です。ヘルプは `:help H` を参照してください。

あくまでも見栄えを調整する機能ですので、別のキーをマップしても大きな問題を起こすことはないでしょう。

j, k, G

もはや説明不要でマッピング不要です。ヘルプは `:help up-down-motions` です。上下移動を細かく制御しつつ高速に行なうのに必須であることに、疑いの余地はありません。

J

次の行との連結であり、重要です。ヘルプは `:help J` です。

同機能の ex コマンドが存在し、一見キーマップしても問題ないように見えますが、1ストロークで実行できる強み、加えて `j` との直交性を考えると推奨したくはありません。

K

カーソルの下の単語を、外部プログラムを用いて検索し、結果を表示する機能です。 ヘルプは `:help K` です。 利用するプログラムをオプション `'keywordprg'` で任意に変更できるほか、 Vim のヘルプを手っ取り早く検索するのにも用います。

Vim の強力な便利さの根幹を為す機能の1つであるため、 キーマップには慎重であるべきところです。しかし逆に言えば、とてつもなく強力な拡張の可能性があるかもしれません。

m

テキスト内の場所を覚えて(マークして)おいて、あとで戻ってくる機能です。ヘルプは `:help mark-motions` です。

マークした場所は、それより前を編集して行番号が変わってしまっても、正しく記憶されています。そのため組み合わせ次第で多彩な編集を可能にしますから、独自キーマップで潰してしまうのは惜しい機能です。しかしマーク機能の拡張という考え方に則るのであれば、何らかの発展の余地があるとも考えられます。

n, N

最後に行った正規表現による検索を繰り返す機能です。
ヘルプは `:help search-commands` です。

Vim 使いであれば当たり前に使う機能のため、 キーマップは実質不可です。

p, P, y, Y

説明不要であろうコピー & ペーストです。ヘルプは `:help copy-move` です。

問答無用でマップ不可、Vim使いならば `p` と `P` を使い分けて当然です。

q

いわゆるキーボードマクロです。ヘルプは `:help complex-repeat` です。簡単に言うと、実際に行ったキー操作を記録して後に再生し、繰り返し同じ操作を実施する、という機能です。

極めて奥深い機能であり、理解して使いこなすのはとてもに難しいのですが、Vim使いである以上は絶対に避けては通れない機能でもあります。

Q

ex モードに切り替えます。ヘルプは `Q` です。

キーマップしても問題ありません。Vimに付属の vimrc_example.vim では `gq` (整形) にマップしています。

u, U

アンドゥ機能です。 ヘルプは `:help undo-commands` です。

おお! この機能をキーマップしてしまうなんて、とんでもない。 あり得ません。

v, V

テキストを視覚的に選択できるビジュアルモード機能です。ヘルプは `:help visual-start` です。

スパルタンVim的には無くても困るべきではない機能ですが、便利であることは間違いないので、まずキーマップしないでしょう。

z

`g` と同じくプレフィックスとして多数の機能が割り当てられています。ヘルプは `:help z` です。

`g` と同じように空きにキーをマップできますが、`ZZ` という取り返しのつかないキーと似ているため、キーマップした際には誤タイプに注意が必要です。

Z

`ZZ` (保存して終了) や `ZQ` (保存せずに終了) といったピーキーな操作があるため、キーマップする際にはくれぐれも慎重に。

その他まとめ

その他、ノーマルモードの簡易なキー一覧は `:help normal-index` に記載されています。

特に `CTRL` との組み合わせでいうと `CTRL-K` と、死角的に `CTRL-[` (`ESC` 相当) が空いており、それぞれ有効なキーマップを考えることができるでしょう。

挿入モード

挿入モードは、テキスト入力という意味でとても重要なモードです。通常のキーは、普通のエディタのようにテキスト入力に用いられ、各種の機能は `CTRL` による修飾により提供されています。本章ではそれらの修飾キーを解説します。

CTRL-A

前回の挿入モードで入力した文字を再入力する機能です。地味に便利ではあるのですが、同じ機能としてより一般的に `<CTRL-R>.` があることを考えると、キーマップするのに適していると言えるでしょう。

CTRL-C

短縮入力 (`:help abbreviations`) を回避して、挿入モードを終了します。通常、挿入モードから抜けるには `ESC` が使えます。短縮入力自体、日本語入力向けにもっと活用されて良さそうな機能なのですが、現在はそうではありません。ゆえに何かしらの機能にマップするのも良いでしょう。

CTRL-D, CTRL-T

カーソルの位置はそのままで、行頭のインデントを調整する機能です。`CTRL-D` で1段階減らし、`CTRL-T` で増やします。プログラマにとっては非常に便利なので、別の機能にマップするべきではないでしょう。

CTRL-E, CTRL-Y

カーソル位置の1つ下(もしくは上)の行にある文字を入力する機能です。1文字だけそうするケースは少なく、繰り返し使うのに連打する必要があり、Vim的ではありません。よって別の機能へのマップもやむなしです。

CTRL-G

標準モードの `g` と同じく、多様な機能のプレフィックスとなっています。下手に上書きする形でキーマップするのではなく、既存機能と直行する形でのキーマップが好ましいでしょう。

CTRL-H

`BS` キーと同じで、カーソルの前の文字を削除します。`BS` は、押すのに左小指を大きく動かす必要のあるキーボードが多いので、こちらで代用するのはスパルタンVim的にとっては基本です。

またマップしてしまうと自動的に `BS` キーもマップされてしまいます。別の機能へマップするのは諦めましょう。

CTRL-I

`Tab` キーと同じです。マップしてしまうと `Tab` キーもマップされてしまいます。通常はマップしません。

CTRL-J, CTRL-M

`Enter` キーと同じです。 `CTRL-M` をマップしてしまうと `Enter` キーもマップされてしまうのに対し、 `CTRL-J` はそうではありません。これを上手く利用して、改行に関係する何かしらの機能をマップするのは悪くないアイデアです。

CTRL-K

Pokémon Go の é を入力するには必須の digraph 機能です。digraph 自体、日本ではあまり活用されていない機能ですが、それゆえに今後の発展が期待できる箇所でもあります。

マップする際には、そのことを考慮した上で行なうのが好ましいでしょう。

CTRL-L, CTRL-Z

ノーマルモードに代えて、 挿入モードを基礎モードとする `'insertmode'` オプション設定時にのみ、意味のある機能が提供されます。

`CTRL-L` はノーマルモードへ移行するという機能を、 `CTRL-Z` は Vim をサスペンドするという機能を持ちます。 `'insertmode'` 自体が通常は使わない機能ですので、マップへの利用は可能です。

CTRL-N, CTRL-P, CTRL-X

Vimにデフォルトで実装されている、多種の補完機能がマップされています。特に `:help i_CTRL-X_index` には目を通すことを強くオススメします。マップは絶対に不可です。

CTRL-O

一時的にノーマルモードへ移行し、1つだけコマンドを実行した後に挿入モードへ戻ってきます。 ESC と挿入モードへの再突入で代用できるとも言えますが、正確には置換モードなどの挿入モードの亜種モードを維持しする機能があるため、完全な代用はできません。

使いこなせれば便利なので、マップは非推奨としたいところですが、そうは言っても活用が難しい機能の1つと言えます。よってマップは任意で、としておきます。

CTRL-Q, CTRL-V

文字コードを指定して入力する機能です。詳細は :help i_CTRL-V_digit を参照してください。マップはすべきではないでしょう。

また CTRL-Q は端末環境によっては、ストップ (CTRL-S) からの復帰に用いられることがあるため、マップができないケースもあります。

CTRL-R

レジスタなどの内容を挿入する機能です。詳細は :help i_CTRL-R を参照してください。

特に :help i_CTRL-R_= は、Vim scriptの実行結果を挿入できるので、その使い方は無限にあります。よってこれをマップするのは論外です。

CTRL-S

CTRL-Q の説明で触れたとおり、 端末環境によってはストップとして使われるので、マップするには不適格です。

CTRL-U

カーソルの前に入力されているテキストを、行頭まで全て削除する機能です。

コマンドラインモード、シェル、他のエディタでも実装・活用される機能であり、直交性の観点からマップする意義の少ないキーです。

CTRL-W

カーソルの前に入力されているテキストを、1単語分削除する機能です。

単語の認識はVimに依存しますので、例によって日本語にとっては非常に貧弱な機能になっていますが、マップするならばそれを補う方向での拡張を検討すべきでしょう。

その他まとめ

ここまで挿入モードの一通りのキーマップを眺めてきましたが、既存機能をほとんど気にせずに独自マップにできるキーは `CTRL-B`, `CTRL-F`, `CTRL-L`, `CTRL-Z` と、意外に多くありました。

`CTRL` + アルファベット以外にも、機能がマップされているキーは多数ありますので、 `:help insert-index` をよく参照してください。

コマンドモード

コマンドモードは、バッチ的な操作をインタラクティブに行えるという点において、Vimにとって最も重要な機能の1つです。そのモードにおけるキー操作は、テキスト入力が優先されるという点で挿入モードと同じですが、修飾キーに伴う機能には共通点と異なる点が混在しています。本章ではそれらのキーを紹介します。

CTRL-A

補完可能なすべての語句を入力する機能です。入力事故も少なくなく、別の機能へのマップもやむなし、と言わざるをえません。

CTRL-B, CTRL-E

カーソルを行頭、行末に移動します。頻繁に利用する機能でもありませんが、マップしてしまうのは惜しいと言えるでしょう。

CTRL-C

挿入モードにおける `CTRL-C` と同様に、 `ESC` と同じようにコマンドモードを終了します。何かしらの機能にマップしても致し方ありません。

CTRL-D

補完可能な語句の一覧を表示します。次の絞り込みのために表示したり、補完の先を見通すために用います。別の機能にマップはすべきではありません。

CTRL-F

コマンドラインウィンドウを開きます。コマンドラインウィンドウ自体、入力履歴をバッファとして扱えるとても便利な機能ですので、わざわざ別の機能にマップする意味はありません。また、オプション `'cedit'` でキーを変更できますが、変える特段の理由もないので、別の機能にマップする必要がありません。

CTRL-H, CTRL-I, CTRL-J, CTRL-M

挿入モードと同様に、それぞれ `BS`, `Tab`, `Enter`, `Enter` と同じ意味です。`CTRL-J` 以外は、 マップしてしまうとそれぞれの対応するキーが使えなくなってしまうので、マップするのは下策と言えます。

なお `Tab` は次の補完の開始および補完候補の選択として機能します。これは `'wildchar'` オプションで変更できますが、 変える意味はありませんし、書くまでもなくマップは非推奨です。

CTRL-K

挿入モードと同じく digraph 機能に割り当てられています。 マップポリシーも同様です。

CTRL-L

補完候補のうち最長の共通する部分を入力する機能に割り当てられています。ファイル名の補完などで、似たプレフィクスのファイルが多いような時に役立ちます。マップすべきではないでしょう。

CTRL-N, CTRL-P

補完候補を選択したり、履歴からの前方一致による補完を開始したりします。後者はカーソルキーの上下でも代用できますが、Vimを使う際には、基本的にカーソルキーへ手を伸ばすべきではありません。よってマップはすべきではありません。

CTRL-Q, CTRL-V

挿入モードと同様に、文字を文字コードを指定して入力する機能です。マップはすべきではないでしょう。

CTRL-R

挿入モードと同様に、レジスタなどの内容を挿入する機能群です。同じく、これをマップするのは論外です。

CTRL-S

挿入モードで言及した通り、端末環境によってはストップとして使われるので、マップするには不適格です。

CTRL-U, CTRL-W

挿入モード同様に、カーソルの前に入力されたテキストを行頭まで全て、もしくは単語1つを削除するコマンドです。どちらもマップする意義は低いと言えます。

CTRL-X

デフォルトでは使われていません。 ただし +kaoriya 版限定で、 現在の

バッファのファイルのあるディレクトリを入力する機能にマップしてあります。

本来は、挿入モードの CTRL-X との対比でより高度な補完のために予約されており、マップには慎重になるべきキーです。

CTRL-Y

特定の場合に、マウスで選んだ範囲をクリップボードへコピーできる機能です。詳細は :help c_CTRL-Y と :help modeless-selection を参照してください。

ウィンドウを超えて見たままの状態をコピーできるのですが、テキスト編集という観点からは意味のある操作に組み込みにくい機能です。マップすることを非難はできません。

CTRL-Z

サスペンド用に予約されています。マップしても問題はありませんが、可能ならば避けるほうが賢明でしょう。

その他まとめ

コマンドラインモードのキーマップを眺めてきました。既存機能を気にせずにマップできるキーとしては CTRL-G, CTRL-O, CTRL-T の3つがありました。また予約済みではありますが、実質的に問題にならないキーとしては CTRL-X, CTRL-Z などがありました。

CTRL + アルファベット以外にも、機能がマップされているキーが多数ありますので、 興味があれば :help ex-edit-index に目を通すと良いでしょう。

まとめ

Vimの3つのモードにおける、基本的なキーマッピングを概観しました。当初 スパルタンVim 的には「Vimの機能を損なわずに大きな影響を与えずにマッピングするのは難しい」という主張をするつもりでした。しかし実際に概観してみると、意外とマッピングの余地があることがわかり、新たな発見をもたらしてくれました。

特に、日本語を取り扱う上でのキーマッピングには、発展の余地が大きいことがわかりました。これには Vim 本体の機能拡張も不可欠でしょう。

一方で、Vimの各モードにおけるキーマップを一覧したい場合には、`:help index` がとても役に立ちます。本書では触れなかった、さらに多くのキーマップがこのヘルプに記載されています。ほとんどの人にとって、それらのキーマップの大半は見たことがないものになるでしょう。本書がそれらを使ってみるキッカケになってくれれば幸いです。

あとがき

実に2年ぶりのスパルタンVimになりました。

間が空いたせいか、今回は随分とスパルタン成分が弱くなってしまったようにも感じます。しかし「スパルタンといえども一切の拡張(キーマップ)を許さないわけではない」ということを示せたという意味では、とても大きな重要な一歩と言えるでしょう。

当初伝えたかったスパルタンVimとしてのメッセージは、「キーマップをする際は、既存機能との整合性を考えろ」でした。しかし期せずして、影響なく利用できるキーが意外に多くあること、また今後どのような方向性の拡張が必要かを、おぼろげながら示す結果となりました。これは純粋に望外の発見です。

今回は Asciidoctor を用いたため、かなり気軽に執筆できました。Rubyだけで PDF まで作成できるというのは実に便利です。 次回はコレを用いて、 既刊の1.0から5.0を一冊にまとめ、 印刷所に印刷製本をお願いしてスパルタンVim総集編その1(仮) としてお届けできたらな、 と考えています。

2016/08/14 村岡太郎 (KoRoN, @kaoriya)