改訂版について

本書スパルタンVim 5.1 は初版であるスパルタンVim 5.0 に修正、 加筆を行った改訂版です。初版はスパルタンと銘打ちながらも、普通のVimユーザーにとっても有用であるとの好評をいただきました。しかしながら幾つかの不備があったことも否めません。そこで本改訂版を制作しました。

不備の一つは初版を出した直後の2016年9月にリリースされたVim 8.0 に伴うものです。 Vim 8.0 ではコマンドラインモードにおけるキーマップに追加が発生し、 初版の一部の記述が 時代遅れのものとなってしまいました。本版はそれに対応する修正を含んでいます。

また初版ではVimを代表する3つのモード、かつ基本的なアルファベットキーに限定して解説していました。しかしVimにはそれ以外のモードや、アルファベット以外のキーにも機能が割り当てられています。本版はそれらについていくらか追記しています。

よりパワーアップしたスパルタンVim 5.1をお楽しみください。

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する機能があります。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の目的は、 デフォルト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検討を促そう、というものです。

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主要なものを紹介します。

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。

特筆すべき点として、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ります。まだ使ったことがない人は是非一度、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

カーソルを画面の上、下、真ん中に移動させる機能です。ヘルプは `:help H` を参照してください。

現在の表示画面という相対的な基準による移動であり、スパルタンとしては不要といえます。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

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, CTRL-R

アンドゥ及び、 それを取り消すリドゥ機能です。 ヘルプは `:help undo-commands` です。

ああ! これ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, CTRL-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。

CTRL-A, CTRL-X

カーソルキーの下の数値を増やしたり、減らしたりする機能です。ヘルプは `:help CTRL-A` です。

工夫次第で、C言語のソースコードを1ストロークで有効/無効を切り替えられたり、簡易な計算機として使えたりと便利な機能なのですが、決定的とまでは言えないのでキーマップは任意としておきます。

CTRL-B/F, CTRL-D/U, CTRL-E/Y

テキストを前後にスクロールします。 `CTRL-B/F` は1画面、 `CTRL-D/U` は半画面、そして `CTRL-E/Y` は1行単位でスクロールします。 ヘルプは `:help scroll-down` 及び `:help scroll-up` です。

キーマップは必要ありませんが、スパルタンVim的には表示という見せかけに依存したスクロールそのものが不要とする考え方もあり、その観点からはキーマップが可能になるという珍しいキーです。

CTRL-C

実行中のコマンドや時間のかかる検索を中止します。端末上での操作との直交性を考慮するとキーマップはすべきではありません。

CTRL-G

現在のファイルの名前やカーソルの位置などの情報を表示します。また似たような機能としてカーソル位置のより詳しい情報が `g CTRL-G` で表示できます。

知っておくとちょっとした時に便利な機能なのですが、使用頻度が高いとは言えません。それを考慮した上でキーマップするのであれば悪くはないでしょう。

CTRL-H, CTRL-J

`h/j` と同じでカーソルを左もしくは下へ動かします。

別の `CTRL` が必要な操作に続けて、 左(下)に移動すべく `h` (`j`) を押した際に勢い余って `CTRL` が押しっぱなしで入力される可能性があります。 その際でもこのキーマップがあれば、意図通りに移動できることになります。

キータイプの精度に自信があればキーマップするのも良いでしょう。

CTRL-I, CTRL-O

ジャンプリストの次/前の要素に移動します。ジャンプリストについては `:help jumplist` を参照してください。

ジャンプリスト自体が非常に有用な機能でありキーマップは不可です。是が非でもジャンプリストを活用してください。

CTRL-N, CTRL-P

CTRL-N/P はカーソルを下もしくは上へ動かします。 他のモードとの直交性を考えれば維持が好ましいですが、キーマップしても全く問題にはなりません。

CTRL-K

デフォルトでは使われていません。好きにキーマップが可能です。

ただし +kaoriya 版においては、 バンドル されている辞書引き プラグイン koron/dicwin-vim のデフォルトリーダーとして割り当てられています。

CTRL-L

画面を再描画します。

シンタックスハイライトの動作が不審な場合や、ターミナルにゴミが残ってしまった場合など、何かと利用する機能であるためキーマップするべきではありません。

CTRL-M

<CR> と同様に、 次の行の先頭の非空白文字に移動します。 この キー を キーマップした場合には、同時に <CR> もマップされます。

いたずらにキーマップするような機能ではありませんが、重要というわけでもありません。キーマップは任意とします。

CTRL-Q, CTRL-S

端末のフロー制御に利用されています。キーマップはできない可能性があります。

CTRL-T

タグリストを遡ります。ヘルプは `:help tag-commands` です。

タグリストはVimの便利さの根幹を成すと言っても過言ではない、非常に有用な機能でありキーマップは不可です。

CTRL-W

各種のウィンドウ操作コマンドのプレフィックスとなっています。ヘルプは `:help window` です。

その重要度を考慮するとキーマップするのは現実的ではありません。

CTRL-Z

Vimをサスペンドしてシェルに戻ります。環境によっては新しいシェルを開始する場合があります。ぜひとも活用して欲しい機能であるため、キーマップは不可とします。

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` にまとめて記載されています。記号なども網羅的に記載されているため、一読をオススメします。死角的に `CTRL-[` (`ESC` 相当)が空いており、キーマップすることも考えられるでしょう。

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-@, CTRL-A

CTRL-A は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。 対して CTRL-@ は挿入後に挿入モードを終了します。どちらも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。

既存のキーマップとしては行を移動する `j/k` 、アンドゥを制御する `u/U` があります。また `CTRL` の押し離しは容易ではありませんので、 `j/k` については `CTRL-J/K` にも同じ機能が割り当てられています。新規のキーマップを追加する際は参考にすると良いでしょう。

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う場合があります。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 `CTRL-X` は多数のサブモードのトリガーのプレフィックスとして機能しています。詳細は `:help i_CTRL-X_index` を参照してください。

マップは基本不可ですが、 `CTRL-X` にはサブモードを自前で追加するようなスタイルのものに成立の余地があるかもしれません。

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、直交性の観点からはマップする意義が弱いキーです。

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。マップの際にはそれを補う方向での拡張を検討すべきでしょう。

CTRL-[

もはや説明不要の `ESC` と同じ機能です。Vimユーザーであれば是が非でも利用するべきでしょう。Touch Barも怖くありません。

CTRL-]

短縮入力を実施する機能です。詳細は `:help abbreviations` を参照してください。短縮入力自体は非キーワードを入力することで自動で起動できるため、通常はこのキーは使わなくても問題ありません。何かしらにうまく利用できそうな雰囲気があります。

CTRL-^

langmap をトグルします。英語キーボードでは `CTRL-6` により入力を代用できます。 langmap についてのヘルプは `:help 'langmap'` です。langmap は日本ではあまり活用されない機能であるため、 キーマップを考えるのは悪くないでしょう。

CTRL-_

日本語では通常は使用しません。気兼ねなくキーマップできます。

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` など、意外に多くありました。

ここで紹介したもの以外にも、機能がマップされているキーは多数ありますので、それらについては `:help insert-index` を参照してください。

コマンドラインモード

コマンドラインモードは、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ライン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-G, CTRL-T

インクリメンタル検索実行時に表示されるマッチ箇所を、次もしくは前のマッチに変更・移動します。Vim 8.0で追加された便利な機能なので別の機能にマップすることは避けるのが望ましいですが、新しいがゆえにこれまでのユーザーにとってはマップして使えなくなってしまっても特に問題のない機能とも言えます。

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。

CTRL-[

もはや説明不要の `ESC` と同じ機能です。Vimユーザーであれば是が非でも利用するべきでしょう。Touch Barも怖くありません。

CTRL-]

短縮入力を実施する機能です。詳細は `:help abbreviations` を参照してください。短縮入力自体は非キーワードを入力することで自動で起動できるため、通常はこのキーは使わなくても問題ありません。何かしらにうまく利用できそうな雰囲気があります。

CTRL-^

langmap をトグルします。英語キーボードでは `CTRL-6` により入力を代用できます。 langmap についてのヘルプは `:help 'langmap'` です。langmap は日本ではあまり活用されない機能であるため、 キーマップを考えるのは悪くないでしょう。

CTRL-_

日本語では通常は使用しません。気兼ねなくキーマップできます。

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-O` だけとなりました。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。

ここで紹介したもの以外にも、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。

ビジュアルモード

ビジュアルモードは視覚的にテキストを選択して操作できるモードです。既存のキーマップが少ないので独自のマップを作りやすく、既存の機能を潰してまでマップする必要性は低いです。そのため特に断らない限りは、既存マップの上書きはすべきではありません。

以下では既存のキーマップのうちアルファベットに関わるものを全て紹介します。

a, i

テキストオブジェクトを指定します。現時点で指定可能なオブジェクトには次のようなものがあります:

```
" ' ( ) < > B W [ ] ` b p s t w { }
```

以上を避け、新たなテキストオブジェクトを定義するのはスパルタン的にも良い考えです。

A, I

矩形ビジュアルモードでは、全てのラインの末尾もしくは先頭にテキストを追加します。

c, C, r, R, s, S

選択した領域を削除し挿入モードを開始します。 大文字の `C R S` は選択領域を行単位で操作します。

d, D, x, X

選択した領域を削除します。 大文字の `D` `X` は選択領域を行単位で操作します。

J

選択した範囲を連結します。

K

選択した範囲を `'keywordprg'` で検索します。

o, O

選択した範囲の反対側へカーソルを移動します。 `o` が選択した範囲全体での反対側なのに対し、 `O` は範囲中のカーソルのある行での反対側へ移動します。

u, U

選択した範囲のアルファベットを小文字もしくは大文字に変換します。

v, V, CTRL-V

ビジュアルモードの選択領域を文字指向、行指向もしくは矩形指向に切り替えます。また既に切り替え先の指向であった場合にはビジュアルモードを終了します。

y, Y

選択範囲をヤンクします。

g

g でプリフィックスされた便利なキーマップが、他のモード同じようにいくつかあります。新規のキーマップを定義する場合にはこれらを避けると良いでしょう。

g CTRL-A, g CTRL-X

選択範囲の数値を漸増的にインクリメントもしくはデクリメントします。連番を生成するのに便利です。

gJ

選択範囲をスペースを挿入せずに連結します。

gq

選択した範囲をフォーマットします。

gv

前回選択したエリアを再選択します。

CTRL-A, CTRL-X

選択した領域の数値をインクリメントもしくはデクリメントします。

CTRL-C

ビジュアルモードを終了します。

CTRL-G

ビジュアルモードと選択モードを切り替えます。選択モードについては `:help Select-mode` を参照してください。

CTRL-H

選択モードのみ: 領域を削除します。

CTRL-O

選択モードのみ: ビジュアルモード に切り替えて 1コマンド実行後に選択モードに戻ります。

その他まとめ

ビジュアルモードは、そもそもスパルタンVimとしては依存すべきではないモードです。しかしまったく使わないというわけではありません。なんでも視覚的に行おうとすることには反対しますが、視覚的であることが本質的な操作については、むしろ積極的に利用すべきです。

その意味において、キーが大量に空いているビジュアルモードは拡張がしやすいのと同時に、どのような拡張を行うかが難しいモードとも言えます。

まとめ

Vimの 4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。

また日本語を取り扱う上でのキーマッピングは、発展の余地が大きいことがわかりました。ただしこれには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

初版から約4ヶ月、スパルタンVimシリーズ初の改訂版5.1です。改訂時にマイナーバージョンを上げて x.1, x.2 とするのは、シリーズ開始当初から決めていましたが、今回やっとお披露目できました。

改訂版に至った要因は、 Vim 8.0 がリリースされたということもありましたが、 なによりも初版 5.0 がスパルタン Vim としては異例な実用度の高さで評価されていた点が大きいです。 改訂内容はまずなによりも Vim 8.0 対応、ノーマルモードでの `CTRL` 装飾についての追記、各モードにおける記号キーに対する追記、ビジュアルモードについての新章、そして全体的に細かなブラッシュアップといった感じになりました。

今回も引き続き Asciidoctor を用いました。ソースを流用できたため、前回同様の楽はできていますが、やはり組版の難しさはいかんともしがたいものがあります。なにせ自分で調整して一行一文字ずつ綺麗に見えているかどうか確認していくわけですから。

初版のあとがきで触れたスパルタンVim総集編は、申し訳ありませんが準備時間不足といいますか、いまいち気乗りしなかったため今回は見送りとなりました。次回はできるといいなぁ。

2016/12/29 村岡太郎 (KoRoN, @kaoriya)