Navigating Emacs windows with win-switch

Christopher Genovese

24 August 2011

Contents

1 Introduction

One wonderful efficiency feature in Emacs is the ability to divide the current view into disjoint rectangular panels (called windows in Emacs terminology) that can be seen and manipulated simultaneously.[1] Taken together, Emacs

[1] There is an unfortunate conflict in terminology between Emacs and traditional window systems. An Emacs *frame* corresponds to a *window* in a traditional window system. An

windows and buffers provide a much more versatile and efficient interface than tabbed browsing or similar designs.

It is not uncommon for people to keep up to 6 windows in a frame. But when you use multiple windows in this way, you may frequently find yourself moving among the windows using `other-window` (`C-x o`) again and again – an unpleasantly large number of keystrokes for so common a maneuver. And because the order of windows in the window list need not relate intuitively to windows' visual positions, moving more efficiently requires context-specific prefix arguments along the way. The tiring outcome is that navigation through a complex window configuration demands many keystrokes and/or nontrivial attention. This package is designed to solve that problem.

While the `windmove` package provides functions for moving intuitively among windows, the natural key bindings for these functions (e.g., the arrow keys with some modifier) require a distant and thus inefficient hand movement. Moreover, one often wants to mix a variety of window-based operations (other-window, previous-window, directional movement, resizing) in rapid succession, which windmove does not easily support.

This package builds on the windmove functionality by defining a command `win-switch-dispatch` that engages a dynamic, transient keyboard override, allowing one to efficiently move among defined windows (and frames) – and even resize, split, delete them – with minimal fuss and effort. When the override is engaged, the movement and resizing commands are bound to simple keys that can be pressed quickly with one hand. The override ends either when the user exits explicitly or after a configurable idle time threshold. The happy outcome is fast and seamless navigation.

2 Installation

To use the package, download `win-switch.el` and place it somewhere in your load path.[2] Then, execute the following code either directly or from in your .emacs file:

```
(require 'win-switch)
```

Emacs *window* is a panel within a frame that displays a *buffer*. There is no direct analog to windows and buffers in standard window systems, though tabbed browsing is perhaps the closest common approach. I will use Emacs terminology in what follows.

[2]Common locations are the system site-lisp directory or the sub-directory `.emacs.d` of the user's home directory. Alternatively, you can load the file directly with the `load-file` command.

```
(global-set-key "\C-xo" 'win-switch-dispatch)
```

or use whatever keybinding you ordinarily have set to `other-window`. Alternatively, you can use one of a variety of predefined configuration commands, as in

```
(require 'win-switch)
(win-switch-setup-keys-ijkl "\C-xo")
```

which has the same effect as the above. In fact, the latter approach is preferred somewhat as it makes it easier to explore other configurations or bindings. See Configuring win-switch below.

At this point, when you execute a window switch (i.e., hit `C-x o`), Emacs will (typically) enter window switching mode allowing you to navigate among the windows (and frames). This is described in more detail in the next section.

3 Using win-switch

When executing the `win-switch-dispatch` command, Emacs's behavior depends on the win-switch parameter settings and the current window configuration. Under the default settings, when there are three or more windows, Emacs will enter window switching mode. This temporarily overrides all key bindings, with selected keys moving among the windows (or frames) and resizing, splitting, or deleting the current window. Window switching lasts until either the user exits explicitly (in one of two ways) or idle time exceeds the threshold in the variable `win-switch-idle-time`. Whenever possible, visual feedback is provided to indicate the window switching state. The following keys are bound by default during window switching mode:

1. Directional Navigation
    - i select the window above the current window.
    - k select the window below the current window.
    - j select the window left of the current window.
    - l select the window right of the current window.
2. Cycling Navigation
    - o cycles forward through the window list in the current frame.

- p cycles backward through the window list in the current frame.
- SPACE cycles among existing frames.

3. Exiting

    - u (and RETURN) exit window switching mode.
    - Any key not bound to a win-switch command exits window switching mode (like u does) and then executes the original function.
    - Waiting idle for more than `win-switch-idle-time` will exit window switching mode (like u does)

- C-M-g is an "emergency" exit[3]

window switching mode in case of an unexpected error during a user defined function or hook (as a customization option) called during window switching mode. This is a paranoid precaution only, and you are very unlikely to need this. Use only as a last resort because it does not handle feedback or other clean up mechanisms. This functionality may be removed in future versions.

1. Resizing

    - I vertically enlarges the current window
    - K vertically shrinks the current window
    - L horizontally enlarges the current window
    - J horizontally shrinks the current window

2. Splitting and Deleting Windows

    - h splits the current window into two equal windows, one above the other. (H is a visual mnemonic for the split.)
    - ; splits the current window into two equal windows, side by side. (: is a visual mnemnoic for the split.)
    - 0 deletes the current window

[3]The emergency exit key causes a no-frills escape from

These keys are chosen to make one-handed navigation fast and easy. It is worth re-emphasizing that any key not bound to a win-switch command exits window switching mode and executes its original function. This is the most common way to exit window switching mode: just do what you want to do in the current window and it will be done. An explicit exit is only required when the first key in the current window is one of the win-switch commands, which should be relatively rare.

By default, window selection wraps around when moving across a frame edge and window switching mode is forgone when there are only two windows. But these features, the key bindings, and other parameters can all be customized, either with the customization facility or with defvar and setter functions, as described in the next section.

4 Configuring win-switch

4.1 Key bindings

The default keybindings are designed for fast and intuitve, one-handed operation, but if desired the key bindings can be easily adjusted or reset. There are several pre-defined key configurations; the key-bindings can be set via the customization mechanism; and there are several functions for modifying the keys associated with particular commands.

4.1.1 Pre-defined key configurations

1. `win-switch-setup-keys-ijkl`

   Sets the default keys centered around the i-j-k-l directional keys. Accepts as argument one or more key-sequences to bind to `win-switch-dispath`, as in

   ```
   (win-switch-setup-keys "\C-xo" "\C-x\C-o")
   ```

2. `win-switch-set-keys-arrow-ctrl`, `win-switch-set-keys-arrow-meta`, `win-switch-set-keys-arrows`

   Sets keys that are centered around the arrow keys, which are used for directional window switching. Control, Meta, or another modifier (respectively) with an arrow key makes a move in that direction and enters window switching mode.

3. `win-switch-set-keys-esdf`

   Left-handed version of the ijkl directional key configuration.

4.1.2 Customize Options

The command keys also can be rebound in groups via the variables with names `win-switch-<name>-keys` where `<name>` can be one of up, down, left, right, next-window, previous-window, enlarge-vertically, shrink-vertically, enlarge-horizontally, shrink-horizontally, other-frame, exit, split-vertically, split-horizontally, delete-window, or emergency-exit. These variables should **not** be set directly, but rather should be set either by the customize mechanism, or by using the functions described below.

4.1.3 Key-setting API

Several functions are available for adjusting the key lists associated with a particular win-switch command. These are `win-switch-add-key`, `win-switch-delete-key`, and `win-switch-set-keys`, which as the names suggest add and delete a key and set the key list, respectively. These functions take a key (or key-list in the latter case) and a command symbol, where the command symbol is one of up, down, left, right, next-window, previous-window, enlarge-vertically, shrink-vertically, enlarge-horizontally, shrink-horizontally, other-frame, exit, split-vertically, split-horizontally, delete-window, or emergency-exit. They are used as follows:

```
(win-switch-add-key    "O" 'previous-window)
(win-switch-delete-key "p" 'previous-window)
(win-switch-set-keys   '(" " "," "m") 'other-frame)
```

Note that the last arguments here are win-switch commands not elisp functions. At least one exit key must always be defined.

The function `win-switch-define-key` is also available for setting general commands in the win-switch keymap, but the other key setting functions and methods are certainly preferred when applicable.

If key bindings are not set by the customize mechanism, they can be set in in the hook `win-switch-load-hook` before loading the package.

4.1.4 Once-Keys

The "command" `win-switch-dispatch-once` is a prefix command/keymap that can be used in place of the standard `win-switch-dispatch` command.

This accepts a single win-switch command, by default using the same key-bindings as standard win-switch (excluding the exit keys), and gives one switch. There are two additional groups of keys `double-next-window` and `double-previous-window` which cycle forward and backward two windows in the list. While not as flexible as window switching mode, the once-dispatch allows easy maneuvering in up to five windows with a single key stroke. The once command keys can be set with either the customize mechanism or the `win-switch-set-once-key` function.

4.1.5 win-switch-dispatch-with

The macro `win-switch-dispatch-with` accepts an elisp command and produces a new command that first executes the given command and then `win-switch-dispatch`. It can be used to create dispatch commands with customized initial behavior. See `win-switch-setup-keys-arrows` for an example.

4.2 Controlling win-switch behavior

Besides key bindings, the most important customization options are the following:

- `win-switch-idle-time`

  Window switching mode exits automatically after Emacs is idle for more than this time. The idle time should be set so that one does not have to either rush or wait. (While explicit exit always works, it is nice to have window-switching mode end on its own at just the right time.) This may require some personalized fiddling to find a comfortable value, though the default should be pretty good.

- `win-switch-window-threshold`

  When the current frame has more than this many windows, `win-switch-dispatch` enters window-switching mode unconditionally; otherwise, it acts like `win-switch-other-window-function` (which is `other-window` by default).

- `win-switch-other-window-first`

  Whether to move to next window in the window list before entering window switching mode. This can be either a boolean value or

a function that returns a boolean function. The latter allows a context sensitive decision; see `win-switch-authors-configuration` for an example.

- `win-switch-wrap-around` (set via `win-switch-set-wrap-around`.)

  If non-nil, directional moves across the edge of the frame wrap around to the other side of the frame (top to bottom, left to right, etc.). This should not be set directly but by the setting function `win-switch-set-wrap-around`.

- `win-switch-other-window-function`

  If non-nil, this should be a function that handles the window switching as does `other-window`. One application of this parameter is when win-switch is used with packages like icicles that remap the other-window function (see `icicle-other-window-or-frame`). If nil, the default, `other-window` is used.

The other customizable parameters control how win-switch mode gives feedback indicating whether window switching is engaged.

- `win-switch-provide-visual-feedback`
- `win-switch-feedback-background-color`
- `win-switch-feedback-foreground-color`
- `win-switch-on-feedback-function`
- `win-switch-off-feedback-function`

The feedback mechanisms are intended to make it salient when window switching mode is on or off and can be customized at several scales. The default method changes the colors on the mode line in the current window during window switching mode (restoring them after), along with transient messages in the echo area.

Finally, three hooks are available to change settings or behavior at load time, when window switching turns on, and when window switching turns off. Extra care should be taken to handle errors properly in the latter two, if they are used.

- `win-switch-load-hook`
- `win-switch-on-hook`
- `win-switch-off-hook`

5 Testing and Bug Reports

The associated package `ws-test.el` in the same repository contains a framework for automated testing of win-switch. Load the package and execute the comand `ws-test-run-all` within emacs, i.e. do `M-x ws-test-run-all`. (There are other ways to run tests in the framework but that is the simplest.) It might be a good idea to manually increase the frame size above the small default initial size before running the tests. A report on the test results will then come up, typically in its own frame. If there are any failures, instructions for reporting tne tests will be given. Please copy the test report and provide as much information as you can about your platform, emacs setup, and ideally your emacs init file. New tests will be added over time.

If when using `win-switch`, you encounter any problems, anomalies, or curiosities, please report them to me at `genovese@cmu.edu`, along with information about your platform, emacs version, and emacs initialization. In addition, feel free to send me feature requests to the same address. In both cases, I would appreciate if you would include "win-switch" on the subject line.

6 Notes

win-switch is not a formal major or minor mode, more of an overriding mode. This started as a way to explore dynamic keybindings, an idea that is generalized considerably in my packages `quick-nav` and `power-keys`. The latter introduces some programming abstractions that can be used to easily install dynamic keymaps of several flavors. I plan to use the `power-keys.el` mechanisms for this package in a later version.