Modifier Keys

Jump to bottom

Nicholas Marriott edited this page on Feb 13 · 21 revisions

Modifier keys

As of tmux 3.5, this document is out of date; tmux now defaults to supporting `extended-keys` in a way similar to xterm.

Terminals support three modifier keys: `Ctrl`, `Meta` (usually `Alt` on modern keyboards) and `Shift`, but which keys support which modifiers and how they are represented to tmux varies between terminals. This document gives an overview of how these keys work and some help on how to troubleshoot them.

What terminal keys look like

Keys are sent to tmux by terminals in three forms:

- Normal ASCII or UTF-8 keys are sent as themselves (`a` is `a`).
- `Ctrl` with ASCII keys are sent using the ASCII control characters (`C-a` is ASCII 1).
- `Meta` prefixes a key with a single ASCII `ESC` character (ASCII 27, sometimes written `^[`, `\033`, `\e` or `\E`). So `M-a` is `^[a`.
- Function keys are sent as a special sequence prefixed by ASCII `ESC`. The exact sequences for different keys varies.

It is important to note:

- The keys available to terminal applications like tmux are not necessarily the same as those available to the terminal itself, for example *X(7)* programs have a much larger range of keys available than can be passed to terminal applications.
- Terminal key sequences are not related to *X(7)* key symbols (used by *xmodmap(1)* or *xev(1)*) or those used by the Linux console.

How tmux describes keys with modifiers

tmux describes keys with modifiers with one of three prefixes:

- `C-` for `Ctrl` keys;
- `M-` for `Meta` keys;
- `S-` for `Shift` keys.

These may be combined, so `Ctrl` and `Meta` and `Left` is `C-M-Left`.

Many keys with `Shift` have an alternative name, tmux uses this if it exists. So there is no `S-a` - it is just in uppercase: `A`. Similarly, `S-Tab` is `BTab`. The `S-` prefix is used for some function keys which have only one form, for example `S-Left` and `S-Right`.

Limitations of `Ctrl` keys

There are only 32 ASCII control characters, so in most terminals there are only 32 `Ctrl` keys:

- `C-@` is ASCII 0;
- `C-a` to `C-z` are ASCII 1 to ASCII 26;
- `C-[`, `C-\`, `C-]`, `C-^` and `C--` are ASCII 27 to 31.

Some of these are used for multiple keys, including:

- `C-@` is also `C-Space`;
- `C-[` is also `Escape`;
- `C-i` is also `Tab`;
- `C-m` is also `Enter`;
- `C--` is also `C-_`;
- `C-^` is also `C-/`.

This means that it is not possible to bind `C-@` and `C-Space` to different things and on most terminals it is not possible to bind some keys like `C-!` or `C-1` at all.

A few terminals have a feature that allows these keys to be used, see this section.

Limitations of `Shift` keys

Most ASCII keys have a `Shift` form marked on the keyboard which is sent when the key is pressed with `Shift`. For example on a UK QWERTY keyboard, pressing `S-1` will send `!`. tmux doesn't know the keyboard layout, so it treats `!` as `!` not `S-1`. There is no way to express the key `S-1` - `!` is used instead.

`Shift` modifiers and the `S-` prefix are mostly reserved for function keys such as `S-F1` or `S-Left`.

Limitations of UTF-8 keys

UTF-8 keys do not have a `Ctrl` or `Shift` form, so they will not work with those modifiers. Because `Meta` works by sending a `^[` prefix, UTF-8 characters can work with a `Meta` modifier.

The escape key

Because the `Escape` key is `^[` which is also the prefix used for `Meta` and for function keys, tmux needs to work out whether a single `^[` is an `Escape` key or part of a longer sequence. It does this using a timer:

- When the `^[` byte is seen, tmux starts the timer;
- If more data comes in before the timer runs out, tmux can work out whether the `^[` is part of a longer sequence;
- Or if the timer expires, the key is `Escape`.

This is why there can be a delay between pressing `Escape` and tmux passing the key on to an application inside. The length of the timer is controlled by the `escape-time` option, the default is 500 milliseconds (half a second).

Common function keys

The sequences that terminals send to tmux for function keys can vary, but for common keys tmux can get the sequences from *terminfo(5)*.

This means that although they may differ between terminals, they usually work. For example `Home` is in the `khom` capability which differs between tmux and *xterm(1)*. The `tput` or `infocmp` commands can be used to inspect *terminfo(5)* capabilities.

```
$ tput -Ttmux khom|cat -v; echo
^[[1~
$ tput -Txterm khom|cat -v; echo
^[OH
```

Because tmux can read `khom`, it can correctly recognise the sequences for this key. In addition, tmux has builtin support for a few common sequences.

Modifiers and function keys

Support for modifiers and function keys, such as `C-F1` or `C-S-Left`, is not always present and these are often the keys that cause most trouble.

xterm(1) offers a descriptive sequence for these keys which many other terminals also use, this includes a number in the key sequence for the modifier, so `C-Left` is `^[[1;5D` where 5 means `Ctrl` and:

- 2 is `Shift`;
- 3 is `Meta`;
- 4 is `Shift` and `Meta`;
- 6 is `Shift` and `Ctrl`;
- 7 is `Meta` and `Ctrl`;
- 8 is `Shift` and `Meta` and `Ctrl`.

These forms are only used for function keys supported by modern terminals - keys which were offered on traditional hardware terminals typically still use their original sequences.

All tmux versions recognise this form of key, and tmux has sent it to application running inside by default since tmux 2.4. In older versions, the `xterm-keys` option must be enabled:

```
set -g xterm-keys on
```

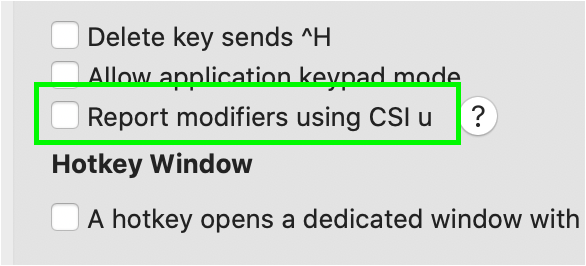
Extended keys

A few terminals have support for extended key sequences, this allows tmux to recognise some keys that are not previously available, such as `C-1` and the other control keys mentioned in this section). See this document for a technical description of the key encoding.

tmux has support for this beginning with tmux 3.2.

For this to work, three things must be in place:

1. The terminal must support it: *xterm(1)*, mintty and iTerm2 currently support this. iTerm2 requires this option to be set in the profile:



2. tmux must be told to turn it on:

```
set -s extended-keys on
```

3. tmux must recognise that the terminal supports it. tmux will automatically detect newer versions of these three terminals, but if it does not then the `terminal-features` option can also be modified to enable it manually:

```
set -as terminal-features 'xterm*:extkeys'
```

Once this feature is enabled, tmux will both recognise extended keys for its own key bindings and forward them to applications inside if they ask for them. For example, sending the escape sequnce to turn it on then running *cat(1)* and pressing `C-1` will show:

```
$ printf '\033[>4;1m'
$ cat
^[[49;5u
```

Why a key might not work

In order for a key to work, two things must be true:

1. tmux and the terminal it is running in must agree on the sequence sent for the key; and
2. tmux and the application running inside must agree on the sequence sent for the key.

The sequence the terminal sends to tmux and the sequence tmux sends to the application inside don't have to be the same - it is tmux's job to translate. But if either tmux and the terminal or tmux and the application do not agree, the key will not be recognised.

Seeing what is sent for a key

The easiest way to see what is being sent for a key is to use *cat(1)* at the shell prompt. For example, running *cat(1)* and pressing `C-Left` in tmux shows:

```
$ cat
^[[1;5D
```

Troubleshooting steps

If a key is not working, the first thing to do is work out if tmux itself is recognizing the key. The easiest way to do this is to try to bind it, for example by running tmux then:

```
$ tmux bind -n TheKey lsk
```

Then if pressing the key shows the `list-keys` output, tmux is recognizing the key. Steps if it does not recognise it are in the next section; and if it does are in the following.

tmux is not recognizing the key

If a tmux key binding doesn't work then:

1. Check something is being sent for the key outside tmux. If nothing appears in *cat(1)* when the key is pressed, the terminal is not sending the key. Perhaps the terminal is using it itself or a window manager is using it instead.
2. For keys with modifiers, make sure the sequence sent for a key outside tmux is different with and without the modifier. If `C-TheKey` and `TheKey` show the same thing with *cat(1)*, tmux can't tell the difference - either the terminal doesn't support the key, or needs additional configuration.
3. Make sure `TERM` is correct. tmux gets some information on keys from `TERM`. Check the terminal documentation to see what `TERM` should be set to outside tmux.
4. If none of these work, open an issue here

tmux is recognizing the key

If a tmux key binding works, but applications inside tmux are not recognizing the key:

1. Check something is shown for the key inside tmux with *cat(1)*. If nothing is shown, make sure there are no bindings for the key in the root table.

2. If *cat(1)* shows some output, make sure `TERM` is correct inside tmux. If it is `screen` or `screen-256color`, try changing to `tmux` or `tmux-256color` if available. Check if they are available with *infocmp(1)*:

```
$ infocmp -x tmux-256color
#       Reconstructed via infocmp from file: /usr/share/terminfo/t/tmux-
256color
tmux-256color|tmux with 256 colors,
...
```

   And if so, set `default-terminal` in `.tmux.conf`:

```
set -g default-terminal tmux-256color
```

3. If `TERM` is correct and *cat(1)* is showing output when the key is pressed, the problem is probably with the application. Check if its documentation has any information on how to configure key bindings.

The number keypad

In most terminals, the number keypad sends either numbers (`1`, `2` and so on) or function keys (`Home`, `Up` and so on). With either of these, tmux cannot tell that the keys are any different from the normal number keys or function keys.

Some terminals additionally allow tmux to put the keypad into "application mode", which allows it to recognise the keys separately so they can be used as key bindings. To check if a terminal supports this, send the `smkx` capability with *tput(1)* then look at the `1` key on the keypad with *cat(1)*:

```
$ tput smkx
$ cat
^[Oq
```

If this shows `^[Oq` then this mode is supported. If it shows something else, it is not supported or the terminal needs additional configuration to enable it.

▸ Recipes

Clone this wiki locally

https://github.com/tmux/tmux.wiki.git