## NAME

tmux — terminal multiplexer

## SYNOPSIS

**tmux** [−2CDlNuVv] [−c *shell-command*] [−f *file*] [−L *socket-name*]
[−S *socket-path*] [−T *features*] [*command* [*flags*]]

## DESCRIPTION

**tmux** is a terminal multiplexer: it enables a number of terminals to be created, accessed, and controlled from a single screen. **tmux** may be detached from a screen and continue running in the background, then later reattached.

When **tmux** is started, it creates a new *session* with a single *window* and displays it on screen. A status line at the bottom of the screen shows information on the current session and is used to enter interactive commands.

A session is a single collection of *pseudo terminals* under the management of **tmux**. Each session has one or more windows linked to it. A window occupies the entire screen and may be split into rectangular panes, each of which is a separate pseudo terminal (the *pty*(4) manual page documents the technical details of pseudo terminals). Any number of **tmux** instances may connect to the same session, and any number of windows may be present in the same session. Once all sessions are killed, **tmux** exits.

Each session is persistent and will survive accidental disconnection (such as *ssh*(1) connection timeout) or intentional detaching (with the `C-b d` key strokes). **tmux** may be reattached using:

```
$ tmux attach
```

In **tmux**, a session is displayed on screen by a *client* and all sessions are managed by a single *server*. The server and each client are separate processes which communicate through a socket in */tmp*.

The options are as follows:

−2
: Force **tmux** to assume the terminal supports 256 colours. This is equivalent to −T *256*.

−C
: Start in control mode (see the “CONTROL MODE” section). Given twice (−CC) disables echo.

−c *shell-command*
: Execute *shell-command* using the default shell. If necessary, the **tmux** server will be started to retrieve the **default-shell** option. This option is for compatibility with *sh*(1) when **tmux** is used as a login shell.

−D
: Do not start the **tmux** server as a daemon. This also turns the **exit-empty** option off. With −D, *command* may not be specified.

−f *file*
: Specify an alternative configuration file. By default, **tmux** loads the system configuration file from */etc/tmux.conf*, if present, then looks for a user configuration file at *~/.tmux.conf* or *$XDG_CONFIG_HOME/tmux/tmux.conf*.

  The configuration file is a set of **tmux** commands which are executed in sequence when the server is first started. **tmux** loads configuration files once when the serv er process has started. The **source-file** command may be used to load a file later.

  **tmux** shows any error messages from commands in configuration files in the first session created, and continues to process the rest of the configuration file.

−L *socket-name*
: **tmux** stores the server socket in a directory under TMUX_TMPDIR or */tmp* if it is unset. The default socket is named *default*. This option allows a different socket name to be specified, allowing several independent **tmux** servers to be run. Unlike −S a full path is not necessary: the sockets are all created in a directory *tmux-UID* under the directory given by TMUX_TMPDIR or in */tmp*. The*tmux-UID* directory is created by **tmux** and

must not be world readable, writable or executable.

If the socket is accidentally removed, the SIGUSR1 signal may be sent to the **tmux** server process to recreate it (note that this will fail if any parent directories are missing).

`-l` Behave as a login shell. This flag currently has no effect and is for compatibility with other shells when using tmux as a login shell.

`-N` Do not start the server even if the command would normally do so (for example **new-session** or **start-server**).

`-S` *socket-path*
Specify a full alternative path to the server socket. If `-S` is specified, the default socket directory is not used and any `-L` flag is ignored.

`-T` *features* Set terminal features for the client. This is a comma-separated list of features. See the **terminal-features** option.

`-u` Write UTF-8 output to the terminal even if the first environment variable of LC_ALL, LC_CTYPE, or LANG that is set does not contain "UTF-8" or "UTF8".

`-V` Report the **tmux** version.

`-v` Request verbose logging. Log messages will be saved into *tmux-client-PID.log* and *tmux-server-PID.log* files in the current directory, where *PID* is the PID of the server or client process. If `-v` is specified twice, an additional *tmux-out-PID.log* file is generated with a copy of everything **tmux** writes to the terminal.

The SIGUSR2 signal may be sent to the **tmux** server process to toggle logging between on (as if `-v` was given) and off.

*command* [*flags*]
This specifies one of a set of commands used to control **tmux**, as described in the following sections. If no commands are specified, the **new-session** command is assumed.

## DEFAULT KEY BINDINGS

**tmux** may be controlled from an attached client by using a key combination of a prefix key, C-b (Ctrl-b) by default, followed by a command key.

The default command key bindings are:

| | |
|---|---|
| C-b | Send the prefix key (C-b) through to the application. |
| C-o | Rotate the panes in the current window forwards. |
| C-z | Suspend the **tmux** client. |
| ! | Break the current pane out of the window. |
| " | Split the current pane into two, top and bottom. |
| # | List all paste buffers. |
| $ | Rename the current session. |
| % | Split the current pane into two, left and right. |
| & | Kill the current window. |
| ' | Prompt for a window index to select. |
| ( | Switch the attached client to the previous session. |
| ) | Switch the attached client to the next session. |
| , | Rename the current window. |
| - | Delete the most recently copied buffer of text. |
| . | Prompt for an index to move the current window. |
| 0 to 9 | Select windows 0 to 9. |
| : | Enter the **tmux** command prompt. |

| | |
|---|---|
| ; | Move to the previously active pane. |
| = | Choose which buffer to paste interactively from a list. |
| ? | List all key bindings. |
| D | Choose a client to detach. |
| L | Switch the attached client back to the last session. |
| [ | Enter copy mode to copy text or view the history. |
| ] | Paste the most recently copied buffer of text. |
| c | Create a new window. |
| d | Detach the current client. |
| f | Prompt to search for text in open windows. |
| i | Display some information about the current window. |
| l | Move to the previously selected window. |
| m | Mark the current pane (see **select-pane** −m). |
| M | Clear the marked pane. |
| n | Change to the next window. |
| o | Select the next pane in the current window. |
| p | Change to the previous window. |
| q | Briefly display pane indexes. |
| r | Force redraw of the attached client. |
| s | Select a new session for the attached client interactively. |
| t | Show the time. |
| w | Choose the current window interactively. |
| x | Kill the current pane. |
| z | Toggle zoom state of the current pane. |
| { | Swap the current pane with the previous pane. |
| } | Swap the current pane with the next pane. |
| ~ | Show previous messages from **tmux**, if any. |
| Page Up | Enter copy mode and scroll one page up. |
| Up, Down<br>Left, Right | Change to the pane above, below, to the left, or to the right of the current pane. |
| M-1 to M-5 | Arrange panes in one of the seven preset layouts: even-horizontal, even-vertical, main-horizontal, main-horizontal-mirrored, main-vertical, main-vertical, or tiled. |
| Space | Arrange the current window in the next preset layout. |
| M-n | Move to the next window with a bell or activity marker. |
| M-o | Rotate the panes in the current window backwards. |
| M-p | Move to the previous window with a bell or activity marker. |
| C-Up, C-Down<br>C-Left, C-Right | Resize the current pane in steps of one cell. |
| M-Up, M-Down<br>M-Left, M-Right | Resize the current pane in steps of five cells. |

Key bindings may be changed with the **bind-key** and **unbind-key** commands.

## COMMAND PARSING AND EXECUTION

**tmux** supports a large number of commands which can be used to control its behaviour. Each command is named and can accept zero or more flags and arguments. They may be bound to a key with the **bind-key** command or run from the shell prompt, a shell script, a configuration file or the command prompt. For example, the same **set-option** command run from the shell prompt, from *~/.tmux.conf* and bound to a key may look like:

```
$ tmux set-option -g status-style bg=cyan

set-option -g status-style bg=cyan
```

```
bind-key C set-option -g status-style bg=cyan
```

Here, the command name is `set-option`, `-g` is a flag and `status-style` and `bg=cyan` are arguments.

**`tmux`** distinguishes between command parsing and execution. In order to execute a command, **`tmux`** needs it to be split up into its name and arguments. This is command parsing. If a command is run from the shell, the shell parses it; from inside **`tmux`** or from a configuration file, **`tmux`** does. Examples of when **`tmux`** parses commands are:

- in a configuration file;
- typed at the command prompt (see **`command-prompt`**);
- given to **`bind-key`**;
- passed as arguments to **`if-shell`** or **`confirm-before`**.

To execute commands, each client has a `command queue`. A global command queue not attached to any client is used on startup for configuration files like *~/.tmux.conf*. Parsed commands added to the queue are executed in order. Some commands, like **`if-shell`** and **`confirm-before`**, parse their argument to create a new command which is inserted immediately after themselves. This means that arguments can be parsed twice or more - once when the parent command (such as **`if-shell`**) is parsed and again when it parses and executes its command. Commands like **`if-shell`**, **`run-shell`** and **`display-panes`** stop execution of subsequent commands on the queue until something happens - **`if-shell`** and **`run-shell`** until a shell command finishes and **`display-panes`** until a key is pressed. For example, the following commands:

```
new-session; new-window
if-shell "true" "split-window"
kill-session
```

Will execute **`new-session`**, **`new-window`**, **`if-shell`**, the shell command *true*(1), **`split-window`** and **`kill-session`** in that order.

The “COMMANDS” section lists the **`tmux`** commands and their arguments.

## PARSING SYNTAX

This section describes the syntax of commands parsed by **`tmux`**, for example in a configuration file or at the command prompt. Note that when commands are entered into the shell, they are parsed by the shell - see for example *ksh*(1) or *csh*(1).

Each command is terminated by a newline or a semicolon (;). Commands separated by semicolons together form a `command sequence` - if a command in the sequence encounters an error, no subsequent commands are executed.

It is recommended that a semicolon used as a command separator should be written as an individual token, for example from *sh*(1):

```
$ tmux neww \; splitw
```

Or:

```
$ tmux neww ';' splitw
```

Or from the tmux command prompt:

```
neww ; splitw
```

However, a trailing semicolon is also interpreted as a command separator, for example in these *sh*(1) commands:

```
$ tmux neww\; splitw
```

Or:

```
$ tmux 'neww;' splitw
```

As in these examples, when running tmux from the shell extra care must be taken to properly quote semicolons:

1. Semicolons that should be interpreted as a command separator should be escaped according to the shell conventions. For *sh*(1) this typically means quoted (such as `neww ';' splitw`) or escaped (such as `neww \\\\; splitw`).

2. Individual semicolons or trailing semicolons that should be interpreted as arguments should be escaped twice: once according to the shell conventions and a second time for **tmux**; for example:

   ```
   $ tmux neww 'foo\\;' bar
   $ tmux neww foo\\\\; bar
   ```

3. Semicolons that are not individual tokens or trailing another token should only be escaped once according to shell conventions; for example:

   ```
   $ tmux neww 'foo-;-bar'
   $ tmux neww foo-\\;-bar
   ```

Comments are marked by the unquoted # character - any remaining text after a comment is ignored until the end of the line.

If the last character of a line is \, the line is joined with the following line (the \ and the newline are completely removed). This is called line continuation and applies both inside and outside quoted strings and in comments, but not inside braces.

Command arguments may be specified as strings surrounded by single (') quotes, double quotes (") or braces ({}). This is required when the argument contains any special character. Single and double quoted strings cannot span multiple lines except with line continuation. Braces can span multiple lines.

Outside of quotes and inside double quotes, these replacements are performed:

- Environment variables preceded by $ are replaced with their value from the global environment (see the “GLOBAL AND SESSION ENVIRONMENT” section).
- A leading ~ or ~user is expanded to the home directory of the current or specified user.
- \uXXXX or \uXXXXXXXX is replaced by the Unicode codepoint corresponding to the given four or eight digit hexadecimal number.
- When preceded (escaped) by a \, the following characters are replaced: \e by the escape character; \r by a carriage return; \n by a newline; and \t by a tab.
- \ooo is replaced by a character of the octal value ooo. Three octal digits are required, for example \001. The largest valid character is \377.
- Any other characters preceded by \ are replaced by themselves (that is, the \ is removed) and are not treated as having any special meaning - so for example \; will not mark a command sequence and \$ will not expand an environment variable.

Braces are parsed as a configuration file (so conditions such as `%if` are processed) and then converted into a string. They are designed to avoid the need for additional escaping when passing a group of **tmux** commands as an argument (for example to **if-shell**). These two examples produce an identical command - note that no escaping is needed when using {}:

```
if-shell true {
    display -p 'brace-dollar-foo: }$foo'
}

if-shell true "display -p 'brace-dollar-foo: }\$foo'"
```

Braces may be enclosed inside braces, for example:

```
bind x if-shell "true" {
    if-shell "true" {
        display "true!"
    }
}
```

Environment variables may be set by using the syntax `name=value`, for example `HOME=/home/user`. Variables set during parsing are added to the global environment. A hidden variable may be set with `%hidden`, for example:

```
%hidden MYVAR=42
```

Hidden variables are not passed to the environment of processes created by tmux. See the "GLOBAL AND SESSION ENVIRONMENT" section.

Commands may be parsed conditionally by surrounding them with `%if`, `%elif`, `%else` and `%endif`. The argument to `%if` and `%elif` is expanded as a format (see "FORMATS") and if it evaluates to false (zero or empty), subsequent text is ignored until the closing `%elif`, `%else` or `%endif`. For example:

```
%if "#{==:#{host},myhost}"
set -g status-style bg=red
%elif "#{==:#{host},myotherhost}"
set -g status-style bg=green
%else
set -g status-style bg=blue
%endif
```

Will change the status line to red if running on `myhost`, green if running on `myotherhost`, or blue if running on another host. Conditionals may be given on one line, for example:

```
%if #{==:#{host},myhost} set -g status-style bg=red %endif
```

## COMMANDS

This section describes the commands supported by **tmux**. Most commands accept the optional `-t` (and sometimes `-s`) argument with one of *target-client*, *target-session*, *target-window*, or *target-pane*. These specify the client, session, window or pane which a command should affect.

*target-client* should be the name of the client, typically the *pty*(4) file to which the client is connected, for example either of */dev/ttyp1* or *ttyp1* for the client attached to */dev/ttyp1*. If no client is specified, **tmux** attempts to work out the client currently in use; if that fails, an error is reported. Clients may be listed with the **list-clients** command.

*target-session* is tried as, in order:

1. A session ID prefixed with a $.
2. An exact name of a session (as listed by the **list-sessions** command).
3. The start of a session name, for example `mysess` would match a session named `mysession`.
4. An *fnmatch*(3) pattern which is matched against the session name.

If the session name is prefixed with an '=', only an exact match is accepted (so `=mysess` will only match exactly `mysess`, not `mysession`).

If a single session is found, it is used as the target session; multiple matches produce an error. If a session is omitted, the current session is used if available; if no current session is available, the most recently used is chosen.

*target-window* (or *src-window* or *dst-window*) specifies a window in the form *session*:*window*. *session* follows the same rules as for *target-session*, and *window* is looked for in order as:

1. A special token, listed below.
2. A window index, for example `mysession:1` is window 1 in session `mysession`.
3. A window ID, such as @1.
4. An exact window name, such as `mysession:mywindow`.
5. The start of a window name, such as `mysession:mywin`.
6. As an *fnmatch*(3) pattern matched against the window name.

Like sessions, a '`=`' prefix will do an exact match only. An empty window name specifies the next unused index if appropriate (for example the **`new-window`** and **`link-window`** commands) otherwise the current window in *session* is chosen.

The following special tokens are available to indicate particular windows. Each has a single-character alternative form.

| **Token** | | **Meaning** |
|---|---|---|
| `{start}` | `^` | The lowest-numbered window |
| `{end}` | `$` | The highest-numbered window |
| `{last}` | `!` | The last (previously current) window |
| `{next}` | `+` | The next window by number |
| `{previous}` | `-` | The previous window by number |

*`target-pane`* (or *`src-pane`* or *`dst-pane`*) may be a pane ID or takes a similar form to *`target-window`* but with the optional addition of a period followed by a pane index or pane ID, for example: `mysession:mywindow.1`. If the pane index is omitted, the currently active pane in the specified window is used. The following special tokens are available for the pane index:

| **Token** | | **Meaning** |
|---|---|---|
| `{last}` | `!` | The last (previously active) pane |
| `{next}` | `+` | The next pane by number |
| `{previous}` | `-` | The previous pane by number |
| `{top}` | | The top pane |
| `{bottom}` | | The bottom pane |
| `{left}` | | The leftmost pane |
| `{right}` | | The rightmost pane |
| `{top-left}` | | The top-left pane |
| `{top-right}` | | The top-right pane |
| `{bottom-left}` | | The bottom-left pane |
| `{bottom-right}` | | The bottom-right pane |
| `{up-of}` | | The pane above the active pane |
| `{down-of}` | | The pane below the active pane |
| `{left-of}` | | The pane to the left of the active pane |
| `{right-of}` | | The pane to the right of the active pane |

The tokens '`+`' and '`-`' may be followed by an offset, for example:

```
select-window -t:+2
```

In addition, *target-session*, *target-window* or *target-pane* may consist entirely of the token `{mouse}` (alternative form '`=`') to specify the session, window or pane where the most recent mouse event occurred (see the "MOUSE SUPPORT" section) or`{marked}` (alternati ve form '`~`') to specify the marked pane (see **`select-pane`** `-m`).

Sessions, window and panes are each numbered with a unique ID; session IDs are prefixed with a '`$`', windows with a '`@`', and panes with a '`%`'. These are unique and are unchanged for the life of the session, window or pane in the **`tmux`** server. The pane ID is passed to the child process of the pane in the`TMUX_PANE` environment variable. IDs may be displayed using the`session_id`, `window_id`, or `pane_id` formats (see the "FORMATS" section) and the**`display-message`**, **`list-sessions`**, **`list-windows`** or

**list-panes** commands.

*shell-command* arguments are *sh*(1) commands. This may be a single argument passed to the shell, for example:

```
new-window 'vi ~/.tmux.conf'
```

Will run:

```
/bin/sh -c 'vi ~/.tmux.conf'
```

Additionally, the **new-window**, **new-session**, **split-window**, **respawn-window** and **respawn-pane** commands allow *shell-command* to be given as multiple arguments and executed directly (without `sh -c`). This can avoid issues with shell quoting. For example:

```
$ tmux new-window vi ~/.tmux.conf
```

Will run *vi*(1) directly without invoking the shell.

*command* [*argument ...*] refers to a **tmux** command, either passed with the command and arguments separately, for example:

```
bind-key F1 set-option status off
```

Or passed as a single string argument in *.tmux.conf*, for example:

```
bind-key F1 { set-option status off }
```

Example **tmux** commands include:

```
refresh-client -t/dev/ttyp2

rename-session -tfirst newname

set-option -wt:0 monitor-activity on

new-window ; split-window -d

bind-key R source-file ~/.tmux.conf \; \
        display-message "source-file done"
```

Or from *sh*(1):

```
$ tmux kill-window -t :1

$ tmux new-window \; split-window -d

$ tmux new-session -d 'vi ~/.tmux.conf' \; split-window -d \; attach
```

## CLIENTS AND SESSIONS

The **tmux** server manages clients, sessions, windows and panes. Clients are attached to sessions to interact with them, either when they are created with the **new-session** command, or later with the **attach-session** command. Each session has one or more windows *linked* into it. Windows may be linked to multiple sessions and are made up of one or more panes, each of which contains a pseudo terminal. Commands for creating, linking and otherwise manipulating windows are covered in the "WINDOWS AND PANES" section.

The following commands are available to manage clients and sessions:

**attach-session** [−dErx] [−c *working-directory*] [−f *flags*] [−t *target-session*]
(alias: **attach**)

If run from outside **tmux**, attach to *target-session* in the current terminal. *target-session* must already exist - to create a new session, see the **new-session** command (with −A to create or attach). If used from inside, switch the currently attached session to

*target-session*. If −d is specified, any other clients attached to the session are detached. If −x is given, send SIGHUP to the parent process of the client as well as detaching the client, typically causing it to exit. −f sets a comma-separated list of client flags. The flags are:

active-pane
: the client has an independent active pane

ignore-size
: the client does not affect the size of other clients

no-output
: the client does not receive pane output in control mode

pause-after=seconds
: output is paused once the pane is *seconds* behind in control mode

read-only
: the client is read-only

wait-exit
: wait for an empty line input before exiting in control mode

A leading '!' turns a flag off if the client is already attached. −r is an alias for −f *read-only,ignore-size*. When a client is read-only, only keys bound to the **detach-client** or **switch-client** commands have any effect. A client with the *active-pane* flag allows the active pane to be selected independently of the window's active pane used by clients without the flag. This only affects the cursor position and commands issued from the client; other features such as hooks and styles continue to use the window's active pane.

If no server is started, **attach-session** will attempt to start it; this will fail unless sessions are created in the configuration file.

The *target-session* rules for **attach-session** are slightly adjusted: if **tmux** needs to select the most recently used session, it will prefer the most recently used *unattached* session.

−c will set the session working directory (used for new windows) to *working-directory*.

If −E is used, the **update-environment** option will not be applied.

**detach-client** [−aP] [−E *shell-command*] [−s *target-session*] [−t *target-client*]
: (alias: **detach**)

Detach the current client if bound to a key, the client specified with −t, or all clients currently attached to the session specified by −s. The −a option kills all but the client given with −t. If −P is given, send SIGHUP to the parent process of the client, typically causing it to exit. With −E, run *shell-command* to replace the client.

**has-session** [−t *target-session*]
: (alias: **has**)

Report an error and exit with 1 if the specified session does not exist. If it does exist, exit with 0.

**kill-server**
: Kill the **tmux** server and clients and destroy all sessions.

**kill-session** [−aC] [−t *target-session*]
: Destroy the given session, closing any windows linked to it and no other sessions, and detaching all clients attached to it. If −a is given, all sessions but the specified one is killed. The −C flag clears alerts (bell, activity, or silence) in all windows linked to the session.

**list-clients** [−F *format*] [−f *filter*] [−t *target-session*]
: (alias: **lsc**)

List all clients attached to the server. −F specifies the format of each line and −f a filter. Only clients for which the filter is true are shown. See the "FORMATS" section. If

*target-session* is specified, list only clients connected to that session.

**list-commands** [−F *format*] [*command*]
(alias: **lscm**)
List the syntax of *command* or - if omitted - of all commands supported by **tmux**.

**list-sessions** [−F *format*] [−f *filter*]
(alias: **ls**)
List all sessions managed by the server. −F specifies the format of each line and −f a filter. Only sessions for which the filter is true are shown. See the "FORMATS" section.

**lock-client** [−t *target-client*]
(alias: **lockc**)
Lock *target-client*, see the **lock-server** command.

**lock-session** [−t *target-session*]
(alias: **locks**)
Lock all clients attached to *target-session*.

**new-session** [−AdDEPX] [−c *start-directory*] [−e *environment*] [−f *flags*] [−F *format*] [−n *window-name*] [−s *session-name*] [−t *group-name*] [−x *width*] [−y *height*] [*shell-command*]
(alias: **new**)
Create a new session with name *session-name*.

The new session is attached to the current terminal unless −d is given. *window-name* and *shell-command* are the name of and shell command to execute in the initial window. With −d, the initial size comes from the global **default-size** option; −x and −y can be used to specify a different size. '-' uses the size of the current client if any. If −x or −y is given, the **default-size** option is set for the session. −f sets a comma-separated list of client flags (see **attach-session**).

If run from a terminal, any *termios*(4) special characters are saved and used for new windows in the new session.

The −A flag makes **new-session** behave like **attach-session** if *session-name* already exists; if −A is given, −D behaves like −d to **attach-session**, and −X behaves like −x to **attach-session**.

If −t is given, it specifies a **session group**. Sessions in the same group share the same set of windows - new windows are linked to all sessions in the group and any windows closed removed from all sessions. The current and previous window and any session options remain independent and any session in a group may be killed without affecting the others. The *group-name* argument may be:

1. the name of an existing group, in which case the new session is added to that group;
2. the name of an existing session - the new session is added to the same group as that session, creating a new group if necessary;
3. the name for a new group containing only the new session.

−n and *shell-command* are invalid if −t is used.

The −P option prints information about the new session after it has been created. By default, it uses the format #{session_name}: but a different format may be specified with −F.

If −E is used, the **update-environment** option will not be applied. −e takes the form VARIABLE=value and sets an environment variable for the newly created session; it may be specified multiple times.

**refresh-client** [−cDLRSU] [−A *pane:state*] [−B *name:what:format*] [−C *size*] [−f *flags*] [−l [*target-pane*]] [−r *pane:report*] [−t *target-client*] [*adjustment*]
(alias: **refresh**)
Refresh the current client if bound to a key, or a single client if one is given with −t. If−S is specified, only update the client's status line.

The −U, −D, −L −R, and −c flags allow the visible portion of a window which is larger than the client to be changed. −U moves the visible part up by *adjustment* rows and −D down, −L left by *adjustment* columns and −R right. −c returns to tracking the cursor automatically. If *adjustment* is omitted, 1 is used. Note that the visible position is a property of the client not of the window, changing the current window in the attached session will reset it.

−C sets the width and height of a control mode client or of a window for a control mode client, *size* must be one of widthxheight or window ID:widthxheight, for example 80x24 or @0:80x24. −A allows a control mode client to trigger actions on a pane. The argument is a pane ID (with leading '%'), a colon, then one of 'on', off, continue or pause. Ifoff, **tmux** will not send output from the pane to the client and if all clients have turned the pane off, will stop reading from the pane. If continue, **tmux** will return to sending output to the pane if it was paused (manually or with the *pause-after* flag). If pause, **tmux** will pause the pane. −A may be given multiple times for different panes.

−B sets a subscription to a format for a control mode client. The argument is split into three items by colons: *name* is a name for the subscription; *what* is a type of item to subscribe to; *format* is the format. After a subscription is added, changes to the format are reported with the **%subscription-changed** notification, at most once a second. If only the name is given, the subscription is removed. *what* may be empty to check the format only for the attached session, or one of: a pane ID such as '%0'; '%∗' for all panes in the attached session; a window ID such as '@0'; or '@∗' for all windows in the attached session.

−f sets a comma-separated list of client flags, see **attach-session**. −r allows a control mode client to provide information about a pane via a report (such as the response to OSC 10). The argument is a pane ID (with a leading '%'), a colon, then a report escape sequence.

−l requests the clipboard from the client using the *xterm*(1) escape sequence. If *target-pane* is given, the clipboard is sent (in encoded form), otherwise it is stored in a new paste buffer.

−L, −R, −U and −D move the visible portion of the window left, right, up or down by *adjustment*, if the window is larger than the client. −c resets so that the position follows the cursor. See the**window-size** option.

**rename-session** [−t *target-session*] *new-name*
(alias: **rename**)
Rename the session to *new-name*.

**server-access** [−adlrw] [*user*]
Change the access or read/write permission of *user*. The user running the**tmux** server (its owner) and the root user cannot be changed and are always permitted access.

−a and −d are used to give or revoke access for the specified user. If the user is already attached, the −d flag causes their clients to be detached.

−r and −w change the permissions for *user*: −r makes their clients read-only and −w writable. −l lists current access permissions.

By default, the access list is empty and **tmux** creates sockets with file system permissions preventing access by any user other than the owner (and root). These permissions must be changed manually. Great care should be taken not to allow access to untrusted users even read-only.

**show-messages** [−JT] [−t *target-client*]
(alias: **showmsgs**)
Show server messages or information. Messages are stored, up to a maximum of the limit set by the *message-limit* server option. −J and −T show debugging information about jobs and terminals.

**source-file** [−Fnqv] [−t *target-pane*] *path* ...
(alias: **source**)
Execute commands from one or more files specified by *path* (which may be *glob*(7) patterns). If −F is present, then *path* is expanded as a format. If −q is given, no error will be returned if *path* does not exist. With −n, the file is parsed but no commands are executed. −v shows the parsed commands and line numbers if possible.

**start-server**
(alias: **start**)
Start the **tmux** server, if not already running, without creating any sessions.

Note that as by default the **tmux** server will exit with no sessions, this is only useful if a session is created in *~/.tmux.conf*, **exit-empty** is turned off, or another command is run as part of the same command sequence. For example:

```
$ tmux start \; show -g
```

**suspend-client** [−t *target-client*]
(alias: **suspendc**)
Suspend a client by sending SIGTSTP (tty stop).

**switch-client** [−ElnprZ] [−c *target-client*] [−t *target-session*] [−T *key-table*]
(alias: **switchc**)
Switch the current session for client *target-client* to *target-session*. As a special case, −t may refer to a pane (a target that contains ':', '.' or '%'), to change session, window and pane. In that case, −Z keeps the window zoomed if it was zoomed. If −l, −n or −p is used, the client is moved to the last, next or previous session respectively. −r toggles the client **read-only** and **ignore-size** flags (see the **attach-session** command).

If −E is used, **update-environment** option will not be applied.

−T sets the client's key table; the next key from the client will be interpreted from *key-table*. This may be used to configure multiple prefix keys, or to bind commands to sequences of keys. For example, to make typing abc run the **list-keys** command:

```
bind-key -Ttable2 c list-keys
bind-key -Ttable1 b switch-client -Ttable2
bind-key -Troot   a switch-client -Ttable1
```

## WINDOWS AND PANES

Each window displayed by **tmux** may be split into one or more *panes*; each pane takes up a certain area of the display and is a separate terminal. A window may be split into panes using the **split-window** command. Windows may be split horizontally (with the −h flag) or vertically. Panes may be resized with the **resize-pane** command (bound to C-Up, C-Down C-Left and C-Right by default), the current pane may be changed with the **select-pane** command and the **rotate-window** and **swap-pane** commands may be used to swap panes without changing their position. Panes are numbered beginning from zero in the order they are created.

By default, a **tmux** pane permits direct access to the terminal contained in the pane. A pane may also be put into one of several modes:

- Copy mode, which permits a section of a window or its history to be copied to a *paste buffer* for later insertion into another window. This mode is entered with the **copy-mode** command, bound to '[' by default. Copied text can be pasted with the **paste-buffer** command, bound

to ‘]’.

- View mode, which is like copy mode but is entered when a command that produces output, such as **list-keys**, is executed from a key binding.
- Choose mode, which allows an item to be chosen from a list. This may be a client, a session or window or pane, or a buffer. This mode is entered with the **choose-buffer**, **choose-client** and **choose-tree** commands.

In copy mode an indicator is displayed in the top-right corner of the pane with the current position and the number of lines in the history.

Commands are sent to copy mode using the −X flag to the **send-keys** command. When a key is pressed, copy mode automatically uses one of two key tables, depending on the **mode-keys** option: **copy-mode** for emacs, or **copy-mode-vi** for vi. Key tables may be viewed with the **list-keys** command.

The following commands are supported in copy mode:

**append-selection**
Append the selection to the top paste buffer.

**append-selection-and-cancel** (vi: A)
Append the selection to the top paste buffer and exit copy mode.

**back-to-indentation** (vi: ^) (emacs: M-m)
Move the cursor back to the indentation.

**begin-selection** (vi: Space) (emacs: C-Space)
Begin selection.

**bottom-line** (vi: L)
Move to the bottom line.

**cancel** (vi: q) (emacs: Escape)
Exit copy mode.

**clear-selection** (vi: Escape) (emacs: C-g)
Clear the current selection.

**copy-end-of-line** [*prefix*]
Copy from the cursor position to the end of the line. *prefix* is used to name the new paste buffer.

**copy-end-of-line-and-cancel** [*prefix*]
Copy from the cursor position and exit copy mode.

**copy-pipe-end-of-line** [*command*] [*prefix*]
Copy from the cursor position to the end of the line and pipe the text to *command*. *prefix* is used to name the new paste buffer.

**copy-pipe-end-of-line-and-cancel** [*command*] [*prefix*]
Same as **copy-pipe-end-of-line** but also exit copy mode.

**copy-line** [*prefix*]
Copy the entire line.

**copy-line-and-cancel** [*prefix*]
Copy the entire line and exit copy mode.

**copy-pipe-line** [*command*] [*prefix*]
Copy the entire line and pipe the text to *command*. *prefix* is used to name the new paste buffer.

**copy-pipe-line-and-cancel** [*command*] [*prefix*]
Same as **copy-pipe-line** but also exit copy mode.

**copy-pipe** [*command*] [*prefix*]
Copy the selection, clear it and pipe its text to *command*. *prefix* is used to name the new paste buffer.

**copy-pipe-no-clear** [*command*] [*prefix*]
Same as **copy-pipe** but do not clear the selection.

**copy-pipe-and-cancel** [*command*] [*prefix*]
Same as **copy-pipe** but also exit copy mode.

**copy-selection** [*prefix*]
Copies the current selection.

**copy-selection-no-clear** [*prefix*]
Same as **copy-selection** but do not clear the selection.

**copy-selection-and-cancel** [*prefix*] (vi: Enter) (emacs: M-w)
Copy the current selection and exit copy mode.

**cursor-down** (vi: j) (emacs: Down)
Move the cursor down.

**cursor-down-and-cancel**
Same as **cursor-down** but also exit copy mode if reaching the bottom.

**cursor-left** (vi: h) (emacs: Left)
Move the cursor left.

**cursor-right** (vi: l) (emacs: Right)
Move the cursor right.

**cursor-up** (vi: k) (emacs: Up)
Move the cursor up.

**end-of-line** (vi: $) (emacs: C-e)
Move the cursor to the end of the line.

**goto-line** *line* (vi: :) (emacs: g)
Move the cursor to a specific line.

**halfpage-down** (vi: C-d) (emacs: M-Down)
Scroll down by half a page.

**halfpage-down-and-cancel**
Same as **halfpage-down** but also exit copy mode if reaching the bottom.

**halfpage-up** (vi: C-u) (emacs: M-Up)
Scroll up by half a page.

**history-bottom** (vi: G) (emacs: M->)
Scroll to the bottom of the history.

**history-top** (vi: g) (emacs: M-<)
Scroll to the top of the history.

**jump-again** (vi: ;) (emacs: ;)
Repeat the last jump.

**jump-backward** *to* (vi: F) (emacs: F)
Jump backwards to the specified text.

**jump-forward** *to* (vi: f) (emacs: f)
Jump forward to the specified text.

**jump-reverse** (vi: ,) (emacs: ,)
Repeat the last jump in the reverse direction (forward becomes backward and backward becomes forward).

**jump-to-backward** *to* (vi: T)
Jump backwards, but one character less, placing the cursor on the character after the target.

**jump-to-forward** *to* (vi: t)
Jump forward, but one character less, placing the cursor on the character before the target.

**jump-to-mark** (vi: M-x) (emacs: M-x)
Jump to the last mark.

**middle-line** (vi: M) (emacs: M-r)
Move to the middle line.

**next-matching-bracket** (vi: %) (emacs: M-C-f)
Move to the next matching bracket.

**next-paragraph** (vi: }) (emacs: M-})
Move to the next paragraph.

**next-prompt** [ −o]
Move to the next prompt.

**next-word** (vi: w)
Move to the next word.

**next-word-end** (vi: e) (emacs: M-f)
Move to the end of the next word.

**next-space** (vi: W)
Same as **next-word** but use a space alone as the word separator.

**next-space-end** (vi: E)
Same as **next-word-end** but use a space alone as the word separator.

**other-end** (vi: o)
Switch at which end of the selection the cursor sits.

**page-down** (vi: C-f) (emacs: PageDown)
Scroll down by one page.

**page-down-and-cancel**
Same as **page-down** but also exit copy mode if reaching the bottom.

**page-up** (vi: C-b) (emacs: PageUp)
Scroll up by one page.

**pipe** [*command*]
Pipe the selected text to *command* and clear the selection.

**pipe-no-clear** [*command*]
Same as **pipe** but do not clear the selection.

**pipe-and-cancel** [*command*] [*prefix*]
Same as **pipe** but also exit copy mode.

**previous-matching-bracket** (emacs: M-C-b)
Move to the previous matching bracket.

**previous-paragraph** (vi: {) (emacs: M-{)
Move to the previous paragraph.

**previous-prompt** [ −o]
Move to the previous prompt.

**previous-word** (vi: b) (emacs: M-b)
Move to the previous word.

**previous-space** (vi: B)
Same as **previous-word** but use a space alone as the word separator.

**rectangle-on**
Turn on rectangle selection mode.

**rectangle-off**
Turn off rectangle selection mode.

**rectangle-toggle** (vi: v) (emacs: R)
Toggle rectangle selection mode.

**refresh-from-pane** (vi: r) (emacs: r)
Refresh the content from the pane.

**scroll-bottom**
Scroll up until the current line is at the bottom while keeping the cursor on that line.

**scroll-down** (vi: C-e) (emacs: C-Down)
Scroll down.

**scroll-down-and-cancel**
Same as **scroll-down** but also exit copy mode if the cursor reaches the bottom.

**scroll-middle** (vi: z)
Scroll so that the current line becomes the middle one while keeping the cursor on that line.

**scroll-top**
Scroll down until the current line is at the top while keeping the cursor on that line.

**scroll-up** (vi: C-y) (emacs: C-Up)
Scroll up.

**search-again** (vi: n) (emacs: n)
Repeat the last search.

**search-backward** *text* (vi: ?)
Search backwards for the specified text.

**search-backward-incremental** *text* (emacs: C-r)
Search backwards incrementally for the specified text. Is expected to be used with the −i flag to the **command-prompt** command.

**search-backward-text** *text*
Search backwards for the specified plain text.

**search-forward** *text* (vi: /)
Search forward for the specified text.

**search-forward-incremental** *text* (emacs: C-s)
Search forward incrementally for the specified text. Is expected to be used with the −i flag to the **command-prompt** command.

**search-forward-text** *text*
Search forward for the specified plain text.

**search-reverse** (vi: N) (emacs: N)
Repeat the last search in the reverse direction (forward becomes backward and backward becomes forward).

**select-line** (vi: V)
Select the current line.

**select-word**
Select the current word.

**set-mark** (vi: X) (emacs: X)
Mark the current line.

**start-of-line** (vi: 0) (emacs: C-a)
Move the cursor to the start of the line.

**stop-selection**
Stop selecting without clearing the current selection.

**toggle-position** (vi: P) (emacs: P)
Toggle the visibility of the position indicator in the top right.

**top-line** (vi: H) (emacs: M-R)
Move to the top line.

The search commands come in several varieties: `search-forward` and `search-backward` search for a regular expression; the `-text` variants search for a plain text string rather than a regular expression; `-incremental` perform an incremental search and expect to be used with the `-i` flag to the **command-prompt** command. `search-again` repeats the last search and `search-reverse` does the same but reverses the direction (forward becomes backward and backward becomes forward).

The `next-prompt` and `previous-prompt` move between shell prompts, but require the shell to emit an escape sequence (\033]133;A\033\\) to tell **tmux** where the prompts are located; if the shell does not do this, these commands will do nothing. The `-o` flag jumps to the be ginning of the command output instead of the shell prompt.

Copy commands may take an optional buffer prefix argument which is used to generate the buffer name (the default is `buffer` so buffers are named `buffer0`, `buffer1` and so on). Pipe commands take a command argument which is the command to which the selected text is piped. `copy-pipe` v ariants also copy the selection. The `-and-cancel` variants of some commands exit copy mode after they have completed (for copy commands) or when the cursor reaches the bottom (for scrolling commands). `-no-clear` variants do not clear the selection.

The next and previous word keys skip over whitespace and treat consecutive runs of either word separators or other letters as words. Word separators can be customized with the *word-separators* session option. Next word moves to the start of the next word, next word end to the end of the next word and previous word to the start of the previous word. The three next and previous space keys work similarly but use a space alone as the word separator. Setting*wor d-separators* to the empty string makes next/previous word equivalent to next/previous space.

The jump commands enable quick movement within a line. For instance, typing '`f`' followed by '`/`' will move the cursor to the next '`/`' character on the current line. A '`;`' will then jump to the next occurrence.

Commands in copy mode may be prefaced by an optional repeat count. With vi key bindings, a prefix is entered using the number keys; with emacs, the Alt (meta) key and a number begins prefix entry.

The synopsis for the **copy-mode** command is:

**copy-mode** [−deHMqu] [−s *src-pane*] [−t *target-pane*]
Enter copy mode. −u also scrolls one page up after entering and −d one page down if already in copy mode. −M begins a mouse drag (only valid if bound to a mouse key binding, see "MOUSE SUPPORT"). −H hides the position indicator in the top right. −q cancels copy mode and any other modes. −s copies from *src-pane* instead of *target-pane*.

−e specifies that scrolling to the bottom of the history (to the visible screen) should exit copy mode. While in copy mode, pressing a key other than those used for scrolling will disable this behaviour. This is intended to allow fast scrolling through a pane's history, for example with:

```
bind PageUp copy-mode -eu
bind PageDown copy-mode -ed
```

A number of preset arrangements of panes are available, these are called layouts. These may be selected with the **select-layout** command or cycled with **next-layout** (bound to Space by default); once a layout is chosen, panes within it may be moved and resized as normal.

The following layouts are supported:

**even-horizontal**
Panes are spread out evenly from left to right across the window.

**even-vertical**
Panes are spread evenly from top to bottom.

**main-horizontal**
A large (main) pane is shown at the top of the window and the remaining panes are spread from left to right in the leftover space at the bottom. Use the *main-pane-height* window option to specify the height of the top pane.

**main-horizontal-mirrored**
The same as **main-horizontal** but mirrored so the main pane is at the bottom of the window.

**main-vertical**
A large (main) pane is shown on the left of the window and the remaining panes are spread from top to bottom in the leftover space on the right. Use the *main-pane-width* window option to specify the width of the left pane.

**main-vertical-mirrored**
The same as **main-vertical** but mirrored so the main pane is on the right of the window.

**tiled** Panes are spread out as evenly as possible over the window in both rows and columns.

In addition, **select-layout** may be used to apply a previously used layout - the **list-windows** command displays the layout of each window in a form suitable for use with **select-layout**. For example:

```
$ tmux list-windows
0: ksh [159x48]
    layout: bb62,159x48,0,0{79x48,0,0,79x48,80,0}
$ tmux select-layout 'bb62,159x48,0,0{79x48,0,0,79x48,80,0}'
```

**tmux** automatically adjusts the size of the layout for the current window size. Note that a layout cannot be applied to a window with more panes than that from which the layout was originally defined.

Commands related to windows and panes are as follows:

**break-pane** [−abdP] [−F *format*] [−n *window-name*] [−s *src-pane*] [−t *dst-window*]
(alias: **breakp**)
Break *src-pane* off from its containing window to make it the only pane in *dst-window*. With −a or −b, the window is moved to the next index after or before (existing windows are moved if necessary). If −d is given, the new window does not become the current window. The −P option prints information about the new window after it has been created. By default, it uses the format #{session_name}:#{window_index}.#{pane_index} but a different

format may be specified with `-F`.

**capture-pane** [`-aAepPqCJN`] [`-b` *buffer-name*] [`-E` *end-line*] [`-S` *start-line*] [`-t` *target-pane*]

(alias: **capturep**)

Capture the contents of a pane. If `-p` is given, the output goes to stdout, otherwise to the buffer specified with `-b` or a new buffer if omitted. If `-a` is given, the alternate screen is used, and the history is not accessible. If no alternate screen exists, an error will be returned unless `-q` is given. If `-e` is given, the output includes escape sequences for text and background attributes. `-C` also escapes non-printable characters as octal \xxx. `-T` ignores trailing positions that do not contain a character. `-N` preserves trailing spaces at each line's end and `-J` preserves trailing spaces and joins any wrapped lines; `-J` implies `-T`. `-P` captures only any output that the pane has received that is the beginning of an as-yet incomplete escape sequence.

`-S` and `-E` specify the starting and ending line numbers, zero is the first line of the visible pane and negative numbers are lines in the history. '`-`' to `-S` is the start of the history and to `-E` the end of the visible pane. The default is to capture only the visible contents of the pane.

**choose-client** [`-NrZ`] [`-F` *format*] [`-f` *filter*] [`-K` *key-format*] [`-O` *sort-order*] [`-t` *target-pane*] [*template*]

Put a pane into client mode, allowing a client to be selected interactively from a list. Each client is shown on one line. A shortcut key is shown on the left in brackets allowing for immediate choice, or the list may be navigated and an item chosen or otherwise manipulated using the keys below. `-Z` zooms the pane. The following keys may be used in client mode:

| Key | Function |
|---|---|
| `Enter` | Choose selected client |
| `Up` | Select previous client |
| `Down` | Select next client |
| `C-s` | Search by name |
| `n` | Repeat last search forwards |
| `N` | Repeat last search backwards |
| `t` | Toggle if client is tagged |
| `T` | Tag no clients |
| `C-t` | Tag all clients |
| `d` | Detach selected client |
| `D` | Detach tagged clients |
| `x` | Detach and HUP selected client |
| `X` | Detach and HUP tagged clients |
| `z` | Suspend selected client |
| `Z` | Suspend tagged clients |
| `f` | Enter a format to filter items |
| `O` | Change sort field |
| `r` | Reverse sort order |
| `v` | Toggle preview |
| `q` | Exit mode |

After a client is chosen, '`%%`' is replaced by the client name in *template* and the result executed as a command. If *template* is not given, "detach-client -t '%%'" is used.

`-O` specifies the initial sort field: one of `name`, `size`, `creation` (time), or `activity` (time). `-r` reverses the sort order. `-f` specifies an initial filter: the filter is a format - if it evaluates to zero, the item in the list is not shown, otherwise it is shown. If a filter would lead to an empty list, it is ignored. `-F` specifies the format for each item in the list and `-K` a format for each shortcut key; both are evaluated once for each line. `-N` starts without the preview. This command works only if at least one client is attached.

**choose-tree** [−GNrswZ] [−F *format*] [−f *filter*] [−K *key-format*] [−O *sort-order*] [−t *target-pane*] [*template*]

Put a pane into tree mode, where a session, window or pane may be chosen interactively from a tree. Each session, window or pane is shown on one line. A shortcut key is shown on the left in brackets allowing for immediate choice, or the tree may be navigated and an item chosen or otherwise manipulated using the keys below. −s starts with sessions collapsed and −w with windows collapsed. −Z zooms the pane. The following keys may be used in tree mode:

| Key | Function |
|---|---|
| Enter | Choose selected item |
| Up | Select previous item |
| Down | Select next item |
| + | Expand selected item |
| - | Collapse selected item |
| M-+ | Expand all items |
| M-- | Collapse all items |
| x | Kill selected item |
| X | Kill tagged items |
| < | Scroll list of previews left |
| > | Scroll list of previews right |
| C-s | Search by name |
| m | Set the marked pane |
| M | Clear the marked pane |
| n | Repeat last search forwards |
| N | Repeat last search backwards |
| t | Toggle if item is tagged |
| T | Tag no items |
| C-t | Tag all items |
| : | Run a command for each tagged item |
| f | Enter a format to filter items |
| H | Jump to the starting pane |
| O | Change sort field |
| r | Reverse sort order |
| v | Toggle preview |
| q | Exit mode |

After a session, window or pane is chosen, the first instance of '%%' and all instances of '%1' are replaced by the target in *template* and the result executed as a command. If *template* is not given, "switch-client -t '%%'" is used.

−O specifies the initial sort field: one of index, name, or time (activity). −r reverses the sort order. −f specifies an initial filter: the filter is a format - if it evaluates to zero, the item in the list is not shown, otherwise it is shown. If a filter would lead to an empty list, it is ignored. −F specifies the format for each item in the tree and −K a format for each shortcut key; both are evaluated once for each line. −N starts without the preview. −G includes all sessions in any session groups in the tree rather than only the first. This command works only if at least one client is attached.

**customize-mode** [−NZ] [−F *format*] [−f *filter*] [−t *target-pane*] [*template*]

Put a pane into customize mode, where options and key bindings may be browsed and modified from a list. Option values in the list are shown for the active pane in the current window. −Z zooms the pane. The following keys may be used in customize mode:

| Key | Function |
|---|---|
| Enter | Set pane, window, session or global option value |

| | |
|---|---|
| `Up` | Select previous item |
| `Down` | Select next item |
| `+` | Expand selected item |
| `-` | Collapse selected item |
| `M-+` | Expand all items |
| `M--` | Collapse all items |
| `s` | Set option value or key attribute |
| `S` | Set global option value |
| `w` | Set window option value, if option is for pane and window |
| `d` | Set an option or key to the default |
| `D` | Set tagged options and tagged keys to the default |
| `u` | Unset an option (set to default value if global) or unbind a key |
| `U` | Unset tagged options and unbind tagged keys |
| `C-s` | Search by name |
| `n` | Repeat last search forwards |
| `N` | Repeat last search backwards |
| `t` | Toggle if item is tagged |
| `T` | Tag no items |
| `C-t` | Tag all items |
| `f` | Enter a format to filter items |
| `v` | Toggle option information |
| `q` | Exit mode |

`-f` specifies an initial filter: the filter is a format - if it evaluates to zero, the item in the list is not shown, otherwise it is shown. If a filter would lead to an empty list, it is ignored. `-F` specifies the format for each item in the tree. `-N` starts without the option information. This command works only if at least one client is attached.

**display-panes** [`-bN`] [`-d` *duration*] [`-t` *target-client*] [*template*]
(alias: **displayp**)

Display a visible indicator of each pane shown by *target-client*. See the **display-panes-colour** and **display-panes-active-colour** session options. The indicator is closed when a key is pressed (unless `-N` is given) or *duration* milliseconds have passed. If `-d` is not given, **display-panes-time** is used. A duration of zero means the indicator stays until a key is pressed. While the indicator is on screen, a pane may be chosen with the '`0`' to '`9`' keys, which will cause *template* to be executed as a command with '`%%`' substituted by the pane ID. The default *template* is "select-pane -t '%%'". With `-b`, other commands are not blocked from running until the indicator is closed.

**find-window** [`-iCNrTZ`] [`-t` *target-pane*] *match-string*
(alias: **findw**)

Search for a *fnmatch*(3) pattern or, with `-r`, regular expression *match-string* in window names, titles, and visible content (but not history). The flags control matching behavior: `-C` matches only visible window contents, `-N` matches only the window name and `-T` matches only the window title. `-i` makes the search ignore case. The default is `-CNT`. `-Z` zooms the pane.

This command works only if at least one client is attached.

**join-pane** [`-bdfhv`] [`-l` *size*] [`-s` *src-pane*] [`-t` *dst-pane*]
(alias: **joinp**)

Like **split-window**, but instead of splitting *dst-pane* and creating a new pane, split it and move *src-pane* into the space. This can be used to reverse **break-pane**. The `-b` option causes *src-pane* to be joined to left of or above *dst-pane*.

If `-s` is omitted and a marked pane is present (see **select-pane** `-m`), the marked pane is used rather than the current pane.

**kill-pane** [−a] [−t *target-pane*]
(alias: **killp**)
Destroy the given pane. If no panes remain in the containing window, it is also destroyed. The −a option kills all but the pane given with −t.

**kill-window** [−a] [−t *target-window*]
(alias: **killw**)
Kill the current window or the window at *target-window*, removing it from any sessions to which it is linked. The −a option kills all b ut the window given with −t.

**last-pane** [−deZ] [−t *target-window*]
(alias: **lastp**)
Select the last (previously selected) pane. −Z keeps the window zoomed if it was zoomed. −e enables or −d disables input to the pane.

**last-window** [−t *target-session*]
(alias: **last**)
Select the last (previously selected) window. If no *target-session* is specified, select the last window of the current session.

**link-window** [−abdk] [−s *src-window*] [−t *dst-window*]
(alias: **linkw**)
Link the window at *src-window* to the specified *dst-window*. If *dst-window* is specified and no such window exists, the *src-window* is linked there. With −a or −b the window is moved to the next index after or before *dst-window* (existing windows are moved if necessary). If −k is given and *dst-window* exists, it is killed, otherwise an error is generated. If −d is given, the newly linked window is not selected.

**list-panes** [−as] [−F *format*] [−f *filter*] [−t *target*]
(alias: **lsp**)
If −a is given, *target* is ignored and all panes on the server are listed. If −s is gi ven, *target* is a session (or the current session). If neither is given, *target* is a window (or the current window). −F specifies the format of each line and −f a filter. Only panes for which the filter is true are shown. See the “FORMATS” section.

**list-windows** [−a] [−F *format*] [−f *filter*] [−t *target-session*]
(alias: **lsw**)
If −a is given, list all windows on the server. Otherwise, list windows in the current session or in *target-session*. −F specifies the format of each line and −f a filter. Only windows for which the filter is true are shown. See the “FORMATS” section.

**move-pane** [−bdfhv] [−l *size*] [−s *src-pane*] [−t *dst-pane*]
(alias: **movep**)
Does the same as **join-pane**.

**move-window** [−abrdk] [−s *src-window*] [−t *dst-window*]
(alias: **movew**)
This is similar to **link-window**, except the window at *src-window* is moved to *dst-window*. With −r, all windows in the session are renumbered in sequential order, respecting the **base-index** option.

**new-window** [−abdkPS] [−c *start-directory*] [−e *environment*] [−F *format*] [−n *window-name*] [−t *target-window*] [*shell-command*]
(alias: **neww**)
Create a new window. With −a or −b, the new window is inserted at the next index after or before the specified *target-window*, moving windows up if necessary; otherwise *target-window* is the new window location.

If `-d` is given, the session does not make the new window the current window. *`target-window`* represents the window to be created; if the target already exists an error is shown, unless the `-k` flag is used, in which case it is destroyed. If `-S` is given and a window named *`window-name`* already exists, it is selected (unless `-d` is also given in which case the command does nothing).

*`shell-command`* is the command to execute. If *`shell-command`* is not specified, the value of the **default-command** option is used. `-c` specifies the working directory in which the new window is created.

When the shell command completes, the window closes. See the **remain-on-exit** option to change this behaviour.

`-e` takes the form `VARIABLE=value` and sets an environment variable for the newly created window; it may be specified multiple times.

The `TERM` environment variable must be set to `screen` or `tmux` for all programs running *inside* **tmux**. New windows will automatically have `TERM=screen` added to their environment, but care must be taken not to reset this in shell start-up files or by the `-e` option.

The `-P` option prints information about the new window after it has been created. By default, it uses the format `#{session_name}:#{window_index}` but a different format may be specified with `-F`.

**next-layout** [`-t` *`target-window`*]
(alias: **nextl**)
Move a window to the next layout and rearrange the panes to fit.

**next-window** [`-a`] [`-t` *`target-session`*]
(alias: **next**)
Move to the next window in the session. If `-a` is used, move to the next window with an alert.

**pipe-pane** [`-IOo`] [`-t` *`target-pane`*] [*`shell-command`*]
(alias: **pipep**)
Pipe output sent by the program in *`target-pane`* to a shell command or vice versa. A pane may only be connected to one command at a time, any existing pipe is closed before *`shell-command`* is executed. The *`shell-command`* string may contain the special character sequences supported by the **status-left** option. If no *`shell-command`* is given, the current pipe (if any) is closed.

`-I` and `-O` specify which of the *`shell-command`* output streams are connected to the pane: with `-I` stdout is connected (so anything *`shell-command`* prints is written to the pane as if it were typed); with `-O` stdin is connected (so any output in the pane is piped to *`shell-command`*). Both may be used together and if neither are specified, `-O` is used.

The `-o` option only opens a new pipe if no previous pipe exists, allowing a pipe to be toggled with a single key, for example:

```
bind-key C-p pipe-pane -o 'cat >>~/output.#I-#P'
```

**previous-layout** [`-t` *`target-window`*]
(alias: **prevl**)
Move to the previous layout in the session.

**previous-window** [`-a`] [`-t` *`target-session`*]
(alias: **prev**)
Move to the previous window in the session. With `-a`, move to the previous window with an alert.

**rename-window** [`-t` *`target-window`*] *`new-name`*
(alias: **renamew**)
Rename the current window, or the window at *`target-window`* if specified, to *`new-name`*.

**resize-pane** [−DLMRTUZ] [−t *target-pane*] [−x *width*] [−y *height*] [*adjustment*]
(alias: **resizep**)

Resize a pane, up, down, left or right by *adjustment* with −U, −D, −L or −R, or to an absolute size with −x or −y. The *adjustment* is given in lines or columns (the default is 1); −x and −y may be a given as a number of lines or columns or followed by '%' for a percentage of the window size (for example -x 10%). With −Z, the active pane is toggled between zoomed (occupying the whole of the window) and unzoomed (its normal position in the layout).

−M begins mouse resizing (only valid if bound to a mouse key binding, see "MOUSE SUPPORT").

−T trims all lines below the current cursor position and moves lines out of the history to replace them.

**resize-window** [−aADLRU] [−t *target-window*] [−x *width*] [−y *height*] [*adjustment*]
(alias: **resizew**)

Resize a window, up, down, left or right by *adjustment* with −U, −D, −L or −R, or to an absolute size with −x or −y. The *adjustment* is given in lines or cells (the default is 1). −A sets the size of the largest session containing the window; −a the size of the smallest. This command will automatically set **window-size** to manual in the window options.

**respawn-pane** [−k] [−c *start-directory*] [−e *environment*] [−t *target-pane*] [*shell-command*]
(alias: **respawnp**)

Reactivate a pane in which the command has exited (see the **remain-on-exit** window option). If *shell-command* is not given, the command used when the pane was created or last respawned is executed. The pane must be already inactive, unless −k is given, in which case any existing command is killed. −c specifies a new working directory for the pane. The −e option has the same meaning as for the **new-window** command.

**respawn-window** [−k] [−c *start-directory*] [−e *environment*] [−t *target-window*] [*shell-command*]
(alias: **respawnw**)

Reactivate a window in which the command has exited (see the **remain-on-exit** window option). If *shell-command* is not given, the command used when the window was created or last respawned is executed. The window must be already inactive, unless −k is given, in which case any existing command is killed. −c specifies a new working directory for the window. The −e option has the same meaning as for the **new-window** command.

**rotate-window** [−DUZ] [−t *target-window*]
(alias: **rotatew**)

Rotate the positions of the panes within a window, either upward (numerically lower) with −U or downward (numerically higher). −Z keeps the window zoomed if it was zoomed.

**select-layout** [−Enop] [−t *target-pane*] [*layout-name*]
(alias: **selectl**)

Choose a specific layout for a window. If *layout-name* is not given, the last preset layout used (if any) is reapplied. −n and −p are equivalent to the **next-layout** and **previous-layout** commands. −o applies the last set layout if possible (undoes the most recent layout change). −E spreads the current pane and any panes next to it out evenly.

**select-pane** [−DdeLlMmRUZ] [−T *title*] [−t *target-pane*]
(alias: **selectp**)

Make pane *target-pane* the active pane in its window. If one of −D, −L, −R, or −U is used, respectively the pane below, to the left, to the right, or above the target pane is used. −Z keeps the window zoomed if it was zoomed. −l is the same as using the **last-pane** command. −e enables or −d disables input to the pane. −T sets the pane title.

`-m` and `-M` are used to set and clear the *marked pane*. There is one marked pane at a time, setting a new marked pane clears the last. The marked pane is the default target for `-s` to **join-pane**, **move-pane**, **swap-pane** and **swap-window**.

**select-window** [`-lnpT`] [`-t` *target-window*]
(alias: **selectw**)
Select the window at *target-window*. `-l`, `-n` and `-p` are equivalent to the **last-window**, **next-window** and **previous-window** commands. If `-T` is given and the selected window is already the current window, the command behaves like **last-window**.

**split-window** [`-bdfhIvPZ`] [`-c` *start-directory*] [`-e` *environment*] [`-l` *size*] [`-t` *target-pane*] [*shell-command*] [`-F` *format*]
(alias: **splitw**)
Create a new pane by splitting *target-pane*: `-h` does a horizontal split and `-v` a vertical split; if neither is specified, `-v` is assumed. The `-l` option specifies the size of the new pane in lines (for vertical split) or in columns (for horizontal split); *size* may be followed by '%' to specify a percentage of the available space. The `-b` option causes the new pane to be created to the left of or above *target-pane*. The `-f` option creates a new pane spanning the full window height (with `-h`) or full window width (with `-v`), instead of splitting the active pane. `-Z` zooms if the window is not zoomed, or keeps it zoomed if already zoomed.

An empty *shell-command* ('') will create a pane with no command running in it. Output can be sent to such a pane with the **display-message** command. The `-I` flag (if *shell-command* is not specified or empty) will create an empty pane and forward any output from stdin to it. For example:

```
$ make 2>&1|tmux splitw -dI &
```

All other options have the same meaning as for the **new-window** command.

**swap-pane** [`-dDUZ`] [`-s` *src-pane*] [`-t` *dst-pane*]
(alias: **swapp**)
Swap two panes. If `-U` is used and no source pane is specified with `-s`, *dst-pane* is swapped with the previous pane (before it numerically); `-D` swaps with the next pane (after it numerically). `-d` instructs **tmux** not to change the active pane and `-Z` keeps the window zoomed if it was zoomed.

If `-s` is omitted and a marked pane is present (see **select-pane** `-m`), the marked pane is used rather than the current pane.

**swap-window** [`-d`] [`-s` *src-window*] [`-t` *dst-window*]
(alias: **swapw**)
This is similar to **link-window**, except the source and destination windows are swapped. It is an error if no window exists at *src-window*. If `-d` is given, the new window does not become the current window.

If `-s` is omitted and a marked pane is present (see **select-pane** `-m`), the window containing the marked pane is used rather than the current window.

**unlink-window** [`-k`] [`-t` *target-window*]
(alias: **unlinkw**)
Unlink *target-window*. Unless `-k` is given, a window may be unlinked only if it is linked to multiple sessions - windows may not be linked to no sessions; if `-k` is specified and the window is linked to only one session, it is unlinked and destroyed.

## KEY BINDINGS

**tmux** allows a command to be bound to most keys, with or without a prefix key. When specifying keys, most represent themselves (for example 'A' to 'Z'). Ctrl keys may be prefixed with 'C-' or '^', Shift keys with 'S-' and Alt (meta) with 'M-'. In addition, the following special key names are accepted: *Up*, *Down*, *Left*, *Right*, *BSpace*, *BTab*, *DC* (Delete), *End*, *Enter*, *Escape*, *F1* to *F12*, *Home*, *IC* (Insert),

*NPage/PageDown/PgDn*, *PPage/PageUp/PgUp*, *Space*, and *Tab*. Note that to bind the ‘"’ or ‘'’ keys, quotation marks are necessary, for example:

```
bind-key '"' split-window
bind-key "'" new-window
```

A command bound to the *Any* key will execute for all keys which do not have a more specific binding.

Commands related to key bindings are as follows:

**bind-key** [−nr] [−N *note*] [−T *key-table*] *key command* [*argument ...*]
(alias: **bind**)
Bind key *key* to *command*. Keys are bound in a key table. By default (without -T), the key is bound in the *prefix* key table. This table is used for keys pressed after the prefix key (for example, by default ‘c’ is bound to **new-window** in the *pr efix* table, so C-b c creates a new window). The *root* table is used for keys pressed without the prefix key: binding ‘c’ to **new-window** in the *root* table (not recommended) means a plain ‘c’ will create a new window. −n is an alias for −T *root*. Keys may also be bound in custom key tables and the **switch-client** −T command used to switch to them from a key binding. The −r flag indicates this k ey may repeat, see the **repeat-time** option. −N attaches a note to the key (shown with **list-keys** −N).

To view the default bindings and possible commands, see the **list-keys** command.

**list-keys** [−1aN] [−P *prefix-string* −T *key-table*] [*key*]
(alias: **lsk**)
List key bindings. There are two forms: the default lists keys as **bind-key** commands; −N lists only keys with attached notes and shows only the key and note for each key.

With the default form, all key tables are listed by default. −T lists only keys in *key-table*.

With the −N form, only k eys in the *root* and *prefix* key tables are listed by default; −T also lists only keys in *key-table*. −P specifies a prefix to print before each key and −1 lists only the first matching key. −a lists the command for keys that do not have a note rather than skipping them.

**send-keys** [−FHKlMRX] [−c *target-client*] [−N *repeat-count*] [−t *target-pane*] *key ...*
(alias: **send**)
Send a key or keys to a window or client. Each argument *key* is the name of the key (such as C-a or NPage) to send; if the string is not recognised as a key, it is sent as a series of characters. If −K is given, keys are sent to *target-client*, so they are looked up in the client’s key table, rather than to *target-pane*. All arguments are sent sequentially from first to last. If no keys are given and the command is bound to a key, then that key is used.

The −l flag disables key name lookup and processes the keys as literal UTF-8 characters. The −H flag expects each key to be a hexadecimal number for an ASCII character.

The −R flag causes the terminal state to be reset.

−M passes through a mouse event (only valid if bound to a mouse key binding, see “MOUSE SUPPORT”).

−X is used to send a command into copy mode - see the “WINDOWS AND PANES” section.−N specifies a repeat count and −F e xpands formats in arguments where appropriate.

**send-prefix** [−2] [−t *target-pane*]
Send the prefix key, or with −2 the secondary prefix k ey, to a window as if it was pressed.

**unbind-key** [−anq] [−T *key-table*] *key*
(alias: **unbind**)
Unbind the command bound to *key*. −n and −T are the same as for **bind-key**. If−a is present, all key bindings are removed. The−q option pre vents errors being returned.

**OPTIONS**

The appearance and behaviour of **tmux** may be modified by changing the value of various options. There are four types of option: *server options*, *session options*, *window options*, and *pane options*.

The **tmux** server has a set of global server options which do not apply to any particular window or session or pane. These are altered with the **set-option** `-s` command, or displayed with the **show-options** `-s` command.

In addition, each individual session may have a set of session options, and there is a separate set of global session options. Sessions which do not have a particular option configured inherit the value from the global session options. Session options are set or unset with the **set-option** command and may be listed with the **show-options** command. The available server and session options are listed under the **set-option** command.

Similarly, a set of window options is attached to each window and a set of pane options to each pane. Pane options inherit from window options. This means any pane option may be set as a window option to apply the option to all panes in the window without the option set, for example these commands will set the background colour to red for all panes except pane 0:

```
set -w window-style bg=red
set -pt:.0 window-style bg=blue
```

There is also a set of global window options from which any unset window or pane options are inherited. Window and pane options are altered with **set-option** `-w` and `-p` commands and displayed with **show-option** `-w` and `-p`.

**tmux** also supports user options which are prefixed with a '`@`'. User options may have any name, so long as they are prefixed with '`@`', and be set to any string. For example:

```
$ tmux set -wq @foo "abc123"
$ tmux show -wv @foo
abc123
```

Commands which set options are as follows:

**set-option** [ `-aFgopqsuUw`] [`-t` *target-pane*] *option value*
(alias: **set**)

Set a pane option with `-p`, a window option with `-w`, a server option with `-s`, otherwise a session option. If the option is not a user option, `-w` or `-s` may be unnecessary - **tmux** will infer the type from the option name, assuming `-w` for pane options. If `-g` is given, the global session or window option is set.

`-F` expands formats in the option value. The `-u` flag unsets an option, so a session inherits the option from the global options (or with `-g`, restores a global option to the default). `-U` unsets an option (like `-u`) but if the option is a pane option also unsets the option on any panes in the window. *value* depends on the option and may be a number, a string, or a flag (on, off, or omitted to toggle).

The `-o` flag prevents setting an option that is already set and the `-q` flag suppresses errors about unknown or ambiguous options.

With `-a`, and if the option expects a string or a style, *value* is appended to the existing setting. For example:

```
set -g status-left "foo"
set -ag status-left "bar"
```

Will result in `foobar`. And:

```
set -g status-style "bg=red"
set -ag status-style "fg=blue"
```

Will result in a red background *and* blue foreground. Without `-a`, the result would be the default background and a blue foreground.

**show-options** [`-AgHpqsvw`] [`-t` *target-pane*] [*option*]
(alias: **show**)
Show the pane options (or a single option if *option* is provided) with `-p`, the window options with `-w`, the server options with `-s`, otherwise the session options. If the option is not a user option, `-w` or `-s` may be unnecessary - **tmux** will infer the type from the option name, assuming `-w` for pane options. Global session or window options are listed if `-g` is used. `-v` shows only the option value, not the name. If `-q` is set, no error will be returned if *option* is unset. `-H` includes hooks (omitted by default). `-A` includes options inherited from a parent set of options, such options are marked with an asterisk.

Available server options are:

**backspace** *key*
Set the key sent by **tmux** for backspace.

**buffer-limit** *number*
Set the number of buffers; as new buffers are added to the top of the stack, old ones are removed from the bottom if necessary to maintain this maximum length.

**command-alias[]** *name=value*
This is an array of custom aliases for commands. If an unknown command matches *name*, it is replaced with *value*. For example, after:

```
set -s command-alias[100] zoom='resize-pane -Z'
```

Using:

```
zoom -t:.1
```

Is equivalent to:

```
resize-pane -Z -t:.1
```

Note that aliases are expanded when a command is parsed rather than when it is executed, so binding an alias with **bind-key** will bind the expanded form.

**copy-command** *shell-command*
Give the command to pipe to if the **copy-pipe** copy mode command is used without arguments.

**default-terminal** *terminal*
Set the default terminal for new windows created in this session - the default value of the `TERM` environment variable. For **tmux** to work correctly, this *must* be set to `screen`, `tmux` or a derivative of them.

**escape-time** *time*
Set the time in milliseconds for which **tmux** waits after an escape is input to determine if it is part of a function or meta key sequences.

**editor** *shell-command*
Set the command used when **tmux** runs an editor.

**exit-empty** [**on** | **off**]
If enabled (the default), the server will exit when there are no active sessions.

**exit-unattached** [**on** | **off**]
If enabled, the server will exit when there are no attached clients.

**extended-keys** [**on** | **off** | **always**]
Controls how modified keys (keys pressed together with Control, Meta, or Shift) are reported. This is the equivalent of the **modifyOtherKeys** *xterm*(1) resource.

When set to **on**, the program inside the pane can request one of two modes: mode 1 which changes the sequence for only keys which lack an existing well-known representation; or mode 2 which changes the sequence for all keys. When set to**always**, modes 1 and 2 can still be requested by applications, but mode 1 will be forced instead of the standard mode. When set to **off**, this feature is disabled and only standard keys are reported.

**tmux** will always request extended keys itself if the terminal supports them. See also the **extkeys** feature for the **terminal-features** option, the **extended-keys-format** option and the **pane_key_mode** variable.

**extended-keys-format** [**csi-u** | **xterm**]
Selects one of the two possible formats for reporting modified keys to applications. This is the equivalent of the **formatOtherKeys** *xterm*(1) resource. For example, C-S-a will be reported as `^[[27;6;65~` when set to **xterm**, and as `^[[65;6u` when set to **csi-u**.

**focus-events** [**on** | **off**]
When enabled, focus events are requested from the terminal if supported and passed through to applications running in **tmux**. Attached clients should be detached and attached again after changing this option.

**history-file** *path*
If not empty, a file to which **tmux** will write command prompt history on exit and load it from on start.

**message-limit** *number*
Set the number of error or information messages to save in the message log for each client.

**prompt-history-limit** *number*
Set the number of history items to save in the history file for each type of command prompt.

**set-clipboard** [**on** | **external** | **off**]
Attempt to set the terminal clipboard content using the *xterm*(1) escape sequence, if there is an *Ms* entry in the *terminfo*(5) description (see the "TERMINFO EXTENSIONS" section).

If set to **on**, **tmux** will both accept the escape sequence to create a buffer and attempt to set the terminal clipboard. If set to **external**, **tmux** will attempt to set the terminal clipboard but ignore attempts by applications to set **tmux** buffers. If**off**, **tmux** will neither accept the clipboard escape sequence nor attempt to set the clipboard.

Note that this feature needs to be enabled in *xterm*(1) by setting the resource:

```
disallowedWindowOps: 20,21,SetXprop
```

Or changing this property from the *xterm*(1) interactive menu when required.

**terminal-features[]** *string*
Set terminal features for terminal types read from *terminfo*(5). **tmux** has a set of named terminal features. Each will apply appropriate changes to the*terminfo*(5) entry in use.

**tmux** can detect features for a few common terminals; this option can be used to easily tell tmux about features supported by terminals it cannot detect. The **terminal-overrides** option allows individual *terminfo*(5) capabilities to be set instead, **terminal-features** is intended for classes of functionality supported in a standard way but not reported by *terminfo*(5). Care must be taken to configure this only with features the terminal actually supports.

This is an array option where each entry is a colon-separated string made up of a terminal type pattern (matched using *fnmatch*(3)) followed by a list of terminal features. The available features are:

256 Supports 256 colours with the SGR escape sequences.

clipboard
Allows setting the system clipboard.

ccolour Allows setting the cursor colour.

cstyle Allows setting the cursor style.

extkeys
: Supports extended keys.

focus Supports focus reporting.

hyperlinks
: Supports OSC 8 hyperlinks.

ignorefkeys
: Ignore function keys from *terminfo*(5) and use the **tmux** internal set only.

margins
: Supports DECSLRM margins.

mouse Supports *xterm*(1) mouse sequences.

osc7 Supports the OSC 7 working directory extension.

overline
: Supports the overline SGR attribute.

rectfill Supports the DECFRA rectangle fill escape sequence.

RGB Supports RGB colour with the SGR escape sequences.

sixel Supports SIXEL graphics.

strikethrough
: Supports the strikethrough SGR escape sequence.

sync Supports synchronized updates.

title Supports *xterm*(1) title setting.

usstyle Allows underscore style and colour to be set.

**terminal-overrides[]** *string*
: Allow terminal descriptions read using *terminfo*(5) to be overridden. Each entry is a colon-separated string made up of a terminal type pattern (matched using *fnmatch*(3)) and a set of *name=value* entries.

  For example, to set the `clear` *terminfo*(5) entry to `\e[H\e[2J` for all terminal types matching `rxvt*`:

```
rxvt*:clear=\e[H\e[2J
```

  The terminal entry value is passed through *strunvis*(3) before interpretation.

**user-keys[]** *key*
: Set list of user-defined key escape sequences. Each item is associated with a key named `User0`, `User1`, and so on.

  For example:

```
set -s user-keys[0] "\e[5;30012~"
bind User0 resize-pane -L 3
```

Available session options are:

**activity-action** [**any** | **none** | **current** | **other**]
: Set action on window activity when **monitor-activity** is on. **any** means activity in any window linked to a session causes a bell or message (depending on **visual-activity**) in the current window of that session, **none** means all activity is ignored (equivalent to

**monitor-activity** being off), **current** means only activity in windows other than the current window are ignored and **other** means activity in the current window is ignored but not those in other windows.

**assume-paste-time** *milliseconds*
If keys are entered faster than one in *milliseconds*, they are assumed to have been pasted rather than typed and **tmux** key bindings are not processed. The default is one millisecond and zero disables.

**base-index** *index*
Set the base index from which an unused index should be searched when a new window is created. The default is zero.

**bell-action** [**any** | **none** | **current** | **other**]
Set action on a bell in a window when **monitor-bell** is on. The values are the same as those for **activity-action**.

**default-command** *shell-command*
Set the command used for new windows (if not specified when the window is created) to *shell-command*, which may be any *sh*(1) command. The default is an empty string, which instructs **tmux** to create a login shell using the value of the **default-shell** option.

**default-shell** *path*
Specify the default shell. This is used as the login shell for new windows when the **default-command** option is set to empty, and must be the full path of the executable. When started **tmux** tries to set a default value from the first suitable of the SHELL environment variable, the shell returned by *getpwuid*(3), or */bin/sh*. This option should be configured when **tmux** is used as a login shell.

**default-size** *XxY*
Set the default size of new windows when the **window-size** option is set to manual or when a session is created with **new-session** -d. The value is the width and height separated by an 'x' character. The default is 80x24.

**destroy-unattached** [**off** | **on** | **keep-last** | **keep-group**]
If **on**, destroy the session after the last client has detached. If **off** (the default), leave the session orphaned. If **keep-last**, destro y the session only if it is in a group and has other sessions in that group. If **keep-group**, destro y the session unless it is in a group and is the only session in that group.

**detach-on-destroy** [**off** | **on** | **no-detached** | **previous** | **next**]
If **on** (the default), the client is detached when the session it is attached to is destroyed. If **off**, the client is switched to the most recently active of the remaining sessions. If **no-detached**, the client is detached only if there are no detached sessions; if detached sessions exist, the client is switched to the most recently active. If **previous** or **next**, the client is switched to the pre vious or next session in alphabetical order.

**display-panes-active-colour** *colour*
Set the colour used by the **display-panes** command to show the indicator for the active pane.

**display-panes-colour** *colour*
Set the colour used by the **display-panes** command to show the indicators for inactive panes.

**display-panes-time** *time*
Set the time in milliseconds for which the indicators shown by the **display-panes** command appear.

**display-time** *time*
Set the amount of time for which status line messages and other on-screen indicators are displayed. If set to 0, messages and indicators are displayed until a key is pressed. *time* is in milliseconds.

**history-limit** *lines*
Set the maximum number of lines held in window history. This setting applies only to new windows - existing window histories are not resized and retain the limit at the point they were created.

**key-table** *key-table*
Set the default key table to *key-table* instead of *root*.

**lock-after-time** *number*
Lock the session (like the **lock-session** command) after *number* seconds of inactivity. The default is not to lock (set to 0).

**lock-command** *shell-command*
Command to run when locking each client. The default is to run *lock*(1) with `-np`.

**menu-style** *style*
Set the menu style. See the "STYLES" section on how to specify *style*. Attributes are ignored.

**menu-selected-style** *style*
Set the selected menu item style. See the "STYLES" section on how to specify *style*. Attributes are ignored.

**menu-border-style** *style*
Set the menu border style. See the "STYLES" section on how to specify *style*. Attributes are ignored.

**menu-border-lines** *type*
Set the type of characters used for drawing menu borders. See **popup-border-lines** for possible values for *border-lines*.

**message-command-style** *style*
Set status line message command style. This is used for the command prompt with *vi*(1) keys when in command mode. For how to specify *style*, see the "STYLES" section.

**message-line** [**0** | **1** | **2** | **3** | **4**]
Set line on which status line messages and the command prompt are shown.

**message-style** *style*
Set status line message style. This is used for messages and for the command prompt. For how to specify *style*, see the "STYLES" section.

**mouse** [**on** | **off**]
If on, **tmux** captures the mouse and allows mouse events to be bound as key bindings. See the "MOUSE SUPPORT" section for details.

**prefix** *key*
Set the key accepted as a prefix key. In addition to the standard keys described under "KEY BINDINGS", **prefix** can be set to the special key `None` to set no prefix.

**prefix2** *key*
Set a secondary key accepted as a prefix key. Like **prefix**, **prefix2** can be set to `None`.

**prefix-timeout** *time*
Set the time in milliseconds for which **tmux** waits after **prefix** is input before dismissing it. Can be set to zero to disable any timeout.

**renumber-windows** [**on** | **off**]
If on, when a window is closed in a session, automatically renumber the other windows in numerical order. This respects the**base-index** option if it has been set. If of f, do not renumber the windows.

**repeat-time** *time*
Allow multiple commands to be entered without pressing the prefix-key again in the specified *time* milliseconds (the default is 500). Whether a key repeats may be set when it is bound using the `-r` flag to **bind-key**. Repeat is enabled for the default keys bound to the **resize-pane** command.

**set-titles** [**on** | **off**]
Attempt to set the client terminal title using the *tsl* and *fsl* *terminfo*(5) entries if they exist. **tmux** automatically sets these to the \e]0;...\007 sequence if the terminal appears to be *xterm*(1). This option is off by default.

**set-titles-string** *string*
String used to set the client terminal title if **set-titles** is on. Formats are expanded, see the "FORMATS" section.

**silence-action** [**any** | **none** | **current** | **other**]
Set action on window silence when **monitor-silence** is on. The values are the same as those for **activity-action**.

**status** [**off** | **on** | **2** | **3** | **4** | **5**]
Show or hide the status line or specify its size. Using **on** gives a status line one row in height; **2**, **3**, **4** or **5** more rows.

**status-format[]** *format*
Specify the format to be used for each line of the status line. The default builds the top status line from the various individual status options below.

**status-interval** *interval*
Update the status line every *interval* seconds. By default, updates will occur every 15 seconds. A setting of zero disables redrawing at interval.

**status-justify** [**left** | **centre** | **right** | **absolute-centre**]
Set the position of the window list in the status line: left, centre or right. centre puts the window list in the relative centre of the available free space; absolute-centre uses the centre of the entire horizontal space.

**status-keys** [**vi** | **emacs**]
Use vi or emacs-style key bindings in the status line, for example at the command prompt. The default is emacs, unless the `VISUAL` or `EDITOR` environment variables are set and contain the string '`vi`'.

**status-left** *string*
Display *string* (by default the session name) to the left of the status line. *string* will be passed through *strftime*(3). Also see the "FORMATS" and "STYLES" sections.

For details on how the names and titles can be set see the "NAMES AND TITLES" section.

Examples are:

```
#(sysctl vm.loadavg)
#[fg=yellow,bold]#(apm -l)%%#[default] [#S]
```

The default is `[#S]` .

**status-left-length** *length*
Set the maximum *length* of the left component of the status line. The default is 10.

**status-left-style** *style*
Set the style of the left part of the status line. For how to specify *style*, see the "STYLES" section.

**status-position** [**top** | **bottom**]
Set the position of the status line.

**status-right** *string*
Display *string* to the right of the status line. By default, the current pane title in double quotes, the date and the time are shown. As with **status-left**, *string* will be passed to *strftime*(3) and character pairs are replaced.

**status-right-length** *length*
Set the maximum *length* of the right component of the status line. The default is 40.

**status-right-style** *style*
Set the style of the right part of the status line. For how to specify *style*, see the "STYLES" section.

**status-style** *style*
Set status line style. For how to specify *style*, see the "STYLES" section.

**update-environment[]** *variable*
Set list of environment variables to be copied into the session environment when a new session is created or an existing session is attached. Any variables that do not exist in the source environment are set to be removed from the session environment (as if `-r` was given to the **set-environment** command).

**visual-activity** [**on** | **off** | **both**]
If on, display a message instead of sending a bell when activity occurs in a window for which the **monitor-activity** window option is enabled. If set to both, a bell and a message are produced.

**visual-bell** [**on** | **off** | **both**]
If on, a message is shown on a bell in a window for which the **monitor-bell** window option is enabled instead of it being passed through to the terminal (which normally makes a sound). If set to both, a bell and a message are produced. Also see the **bell-action** option.

**visual-silence** [**on** | **off** | **both**]
If **monitor-silence** is enabled, prints a message after the interval has expired on a given window instead of sending a bell. If set to both, a bell and a message are produced.

**word-separators** *string*
Sets the session's conception of what characters are considered word separators, for the purposes of the next and previous word commands in copy mode.

Available window options are:

**aggressive-resize** [**on** | **off**]
Aggressively resize the chosen window. This means that **tmux** will resize the windo w to the size of the smallest or largest session (see the **window-size** option) for which it is the current window, rather than the session to which it is attached. The window may resize when the current window is changed on another session; this option is good for full-screen programs which support `SIGWINCH` and poor for interactive programs such as shells.

**automatic-rename** [**on** | **off**]
Control automatic window renaming. When this setting is enabled, **tmux** will rename the window automatically using the format specified by **automatic-rename-format**. This flag is automatically disabled for an individual window when a name is specified at creation with **new-window** or **new-session**, or later with **rename-window**, or with a terminal escape sequence. It may be switched off globally with:

```
set-option -wg automatic-rename off
```

**automatic-rename-format** *format*
The format (see “FORMATS”) used when the **automatic-rename** option is enabled.

**clock-mode-colour** *colour*
Set clock colour.

**clock-mode-style** [**12** | **24**]
Set clock hour format.

**fill-character** *character*
Set the character used to fill areas of the terminal unused by a window.

**main-pane-height** *height*
**main-pane-width** *width*
Set the width or height of the main (left or top) pane in the **main-horizontal**, **main-horizontal-mirrored**, **main-vertical**, or **main-vertical-mirrored** layouts. If suffixed by ‘%’, this is a percentage of the window size.

**copy-mode-match-style** *style*
Set the style of search matches in copy mode. For how to specify *style*, see the “STYLES” section.

**copy-mode-mark-style** *style*
Set the style of the line containing the mark in copy mode. For how to specify *style*, see the “STYLES” section.

**copy-mode-current-match-style** *style*
Set the style of the current search match in copy mode. For how to specify *style*, see the “STYLES” section.

**mode-keys** [**vi** | **emacs**]
Use vi or emacs-style key bindings in copy mode. The default is emacs, unless VISUAL or EDITOR contains ‘vi’.

**mode-style** *style*
Set window modes style. For how to specify *style*, see the “STYLES” section.

**monitor-activity** [**on** | **off**]
Monitor for activity in the window. Windows with activity are highlighted in the status line.

**monitor-bell** [**on** | **off**]
Monitor for a bell in the window. Windows with a bell are highlighted in the status line.

**monitor-silence** [**interval**]
Monitor for silence (no activity) in the window within **interval** seconds. Windows that have been silent for the interval are highlighted in the status line. An interval of zero disables the monitoring.

**other-pane-height** *height*
Set the height of the other panes (not the main pane) in the **main-horizontal** and **main-horizontal-mirrored** layouts. If this option is set to 0 (the default), it will have no effect. If both the**main-pane-height** and **other-pane-height** options are set, the main pane will grow taller to make the other panes the specified height, but will never shrink to do so. If suffixed by ‘%’, this is a percentage of the window size.

**other-pane-width** *width*
Like **other-pane-height**, but set the width of other panes in the **main-vertical** and **main-vertical-mirrored** layouts.

**pane-active-border-style** *style*
Set the pane border style for the currently active pane. For how to specify *style*, see the “STYLES” section. Attributes are ignored.

**pane-base-index** *index*
Like **base-index**, but set the starting index for pane numbers.

**pane-border-format** *format*
Set the text shown in pane border status lines.

**pane-border-indicators** [**off** | **colour** | **arrows** | **both**]
Indicate active pane by colouring only half of the border in windows with exactly two panes, by displaying arrow markers, by drawing both or neither.

**pane-border-lines** *type*
Set the type of characters used for drawing pane borders. *type* may be one of:

single single lines using ACS or UTF-8 characters

double double lines using UTF-8 characters

heavy heavy lines using UTF-8 characters

simple simple ASCII characters

number
the pane number

`double` and `heavy` will fall back to standard ACS line drawing when UTF-8 is not supported.

**pane-border-status** [**off** | **top** | **bottom**]
Turn pane border status lines off or set their position.

**pane-border-style** *style*
Set the pane border style for panes aside from the active pane. For how to specify *style*, see the "STYLES" section. Attributes are ignored.

**popup-style** *style*
Set the popup style. See the "STYLES" section on how to specify *style*. Attributes are ignored.

**popup-border-style** *style*
Set the popup border style. See the "STYLES" section on how to specify *style*. Attributes are ignored.

**popup-border-lines** *type*
Set the type of characters used for drawing popup borders. *type* may be one of:

single single lines using ACS or UTF-8 characters (default)

rounded
variation of single with rounded corners using UTF-8 characters

double double lines using UTF-8 characters

heavy heavy lines using UTF-8 characters

simple simple ASCII characters

padded simple ASCII space character

none no border

`double` and `heavy` will fall back to standard ACS line drawing when UTF-8 is not supported.

**window-status-activity-style** *style*
Set status line style for windows with an activity alert. For how to specify *style*, see the "STYLES" section.

**window-status-bell-style** *style*
Set status line style for windows with a bell alert. For how to specify *style*, see the "STYLES" section.

**window-status-current-format** *string*
Like *window-status-format*, but is the format used when the window is the current window.

**window-status-current-style** *style*
Set status line style for the currently active window. For how to specify *style*, see the "STYLES" section.

**window-status-format** *string*
Set the format in which the window is displayed in the status line window list. See the "FORMATS" and "STYLES" sections.

**window-status-last-style** *style*
Set status line style for the last active window. For how to specify *style*, see the "STYLES" section.

**window-status-separator** *string*
Sets the separator drawn between windows in the status line. The default is a single space character.

**window-status-style** *style*
Set status line style for a single window. For how to specify *style*, see the "STYLES" section.

**window-size** *largest* | *smallest* | *manual* | *latest*
Configure how **tmux** determines the window size. If set to *largest*, the size of the largest attached session is used; if *smallest*, the size of the smallest. If *manual*, the size of a new window is set from the **default-size** option and windows are resized automatically. With *latest*, **tmux** uses the size of the client that had the most recent activity. See also the **resize-window** command and the **aggressive-resize** option.

**wrap-search** [**on** | **off**]
If this option is set, searches will wrap around the end of the pane contents. The default is on.

Available pane options are:

**allow-passthrough** [**on** | **off** | **all**]
Allow programs in the pane to bypass **tmux** using a terminal escape sequence (\ePtmux;...\e\\). If set to **on**, passthrough sequences will be allowed only if the pane is visible. If set to **all**, they will be allowed even if the pane is invisible.

**allow-rename** [**on** | **off**]
Allow programs in the pane to change the window name using a terminal escape sequence (\ek...\e\\).

**allow-set-title** [**on** | **off**]
Allow programs in the pane to change the title using the terminal escape sequences (\e]2;...\e\\ or \e]0;...\e\\).

**alternate-screen** [**on** | **off**]
This option configures whether programs running inside the pane may use the terminal alternate screen feature, which allows the *smcup* and *rmcup terminfo*(5) capabilities. The alternate screen feature preserves the contents of the window when an interactive application starts and restores it on exit, so that any output visible before the application starts reappears unchanged after it exits.

**cursor-colour** *colour*
Set the colour of the cursor.

**pane-colours[]** *colour*
The default colour palette. Each entry in the array defines the colour **tmux** uses when the colour with that index is requested. The index may be from zero to 255.

**cursor-style** *style*
Set the style of the cursor. Available styles are: **default**, **blinking-block**, **block**, **blinking-underline**, **underline**, **blinking-bar**, **bar**.

**remain-on-exit** [**on** | **off** | **failed**]
A pane with this flag set is not destroyed when the program running in it exits. If set to **failed**, then only when the program exit status is not zero. The pane may be reactivated with the **respawn-pane** command.

**remain-on-exit-format** *string*
Set the text shown at the bottom of exited panes when **remain-on-exit** is enabled.

**scroll-on-clear** [**on** | **off**]
When the entire screen is cleared and this option is on, scroll the contents of the screen into history before clearing it.

**synchronize-panes** [**on** | **off**]
Duplicate input to all other panes in the same window where this option is also on (only for panes that are not in any mode).

**window-active-style** *style*
Set the pane style when it is the active pane. For how to specify *style*, see the "STYLES" section.

**window-style** *style*
Set the pane style. For how to specify *style*, see the "STYLES" section.

## HOOKS

**tmux** allows commands to run on various triggers, called *hooks*. Most**tmux** commands ha ve an *after* hook and there are a number of hooks not associated with commands.

Hooks are stored as array options, members of the array are executed in order when the hook is triggered. Like options different hooks may be global or belong to a session, window or pane. Hooks may be configured with the **set-hook** or **set-option** commands and displayed with **show-hooks** or **show-options** −H. The following two commands are equivalent:

```
set-hook -g pane-mode-changed[42] 'set -g status-left-style bg=red'
set-option -g pane-mode-changed[42] 'set -g status-left-style bg=red'
```

Setting a hook without specifying an array index clears the hook and sets the first member of the array.

A command's after hook is run after it completes, except when the command is run as part of a hook itself. They are named with an `after-` prefix. For example, the following command adds a hook to select the even-vertical layout after every **split-window**:

```
set-hook -g after-split-window "selectl even-vertical"
```

If a command fails, the `command-error` hook will be fired. For example, this could be used to write to a log file:

```
set-hook -g command-error "run-shell \"echo 'a tmux command failed' >>/tmp/l
```

All the notifications listed in the "CONTROL MODE" section are hooks (without any arguments), except **%exit**. The following additional hooks are available:

| | |
|---|---|
| alert-activity | Run when a window has activity. See**monitor-activity**. |
| alert-bell | Run when a window has received a bell. See **monitor-bell**. |
| alert-silence | Run when a window has been silent. See **monitor-silence**. |
| client-active | Run when a client becomes the latest active client of its session. |
| client-attached | Run when a client is attached. |
| client-detached | Run when a client is detached |
| client-focus-in | Run when focus enters a client |

| | |
|---|---|
| client-focus-out | Run when focus exits a client |
| client-resized | Run when a client is resized. |
| client-session-changed | Run when a client's attached session is changed. |
| command-error | Run when a command fails. |
| pane-died | Run when the program running in a pane exits, but **remain-on-exit** is on so the pane has not closed. |
| pane-exited | Run when the program running in a pane exits. |
| pane-focus-in | Run when the focus enters a pane, if the **focus-events** option is on. |
| pane-focus-out | Run when the focus exits a pane, if the **focus-events** option is on. |
| pane-set-clipboard | Run when the terminal clipboard is set using the *xterm*(1) escape sequence. |
| session-created | Run when a new session created. |
| session-closed | Run when a session closed. |
| session-renamed | Run when a session is renamed. |
| window-linked | Run when a window is linked into a session. |
| window-renamed | Run when a window is renamed. |
| window-resized | Run when a window is resized. This may be after the *client-resized* hook is run. |
| window-unlinked | Run when a window is unlinked from a session. |

Hooks are managed with these commands:

**set-hook** [−agpRuw] [−t *target-pane*] *hook-name command*
Without −R, sets (or with −u unsets) hook *hook-name* to *command*. The flags are the same as for **set-option**.

With −R, run *hook-name* immediately.

**show-hooks** [−gpw] [−t *target-pane*]
Shows hooks. The flags are the same as for **show-options**.

## MOUSE SUPPORT

If the **mouse** option is on (the default is off), **tmux** allows mouse events to be bound as keys. The name of each key is made up of a mouse event (such as MouseUp1) and a location suffix, one of the following:

| | |
|---|---|
| Pane | the contents of a pane |
| Border | a pane border |
| Status | the status line window list |
| StatusLeft | the left part of the status line |
| StatusRight | the right part of the status line |
| StatusDefault | any other part of the status line |

The following mouse events are available:

| | | | |
|---|---|---|---|
| WheelUp | WheelDown | | |
| MouseDown1 | MouseUp1 | MouseDrag1 | MouseDragEnd1 |
| MouseDown2 | MouseUp2 | MouseDrag2 | MouseDragEnd2 |
| MouseDown3 | MouseUp3 | MouseDrag3 | MouseDragEnd3 |

| | | |
|---|---|---|
| `SecondClick1` | SecondClick2 | SecondClick3 |
| `DoubleClick1` | DoubleClick2 | DoubleClick3 |
| `TripleClick1` | TripleClick2 | TripleClick3 |

The `SecondClick` events are fired for the second click of a double click, even if there may be a third click which will fire `TripleClick` instead of `DoubleClick`.

Each should be suffixed with a location, for example `MouseDown1Status`.

The special token `{mouse}` or '`=`' may be used as *`target-window`* or *`target-pane`* in commands bound to mouse key bindings. It resolves to the window or pane over which the mouse event took place (for example, the window in the status line over which button 1 was released for a `MouseUp1Status` binding, or the pane over which the wheel was scrolled for a `WheelDownPane` binding).

The **`send-keys`** `-M` flag may be used to forward a mouse event to a pane.

The default key bindings allow the mouse to be used to select and resize panes, to copy text and to change window using the status line. These take effect if the **`mouse`** option is turned on.

**FORMATS**

Certain commands accept the `-F` flag with a *`format`* argument. This is a string which controls the output format of the command. Format variables are enclosed in '`#{`' and '`}`', for example `#{session_name}`. The possible variables are listed in the table below, or the name of a **`tmux`** option may be used for an option's value. Some variables have a shorter alias such as '`#S`'; '`##`' is replaced by a single '`#`', '`#,`' by a '`,`' and '`#}`' by a '`}`'.

Conditionals are available by prefixing with '`?`' and separating two alternatives with a comma; if the specified variable exists and is not zero, the first alternative is chosen, otherwise the second is used. For example `#{?session_attached,attached,not attached}`will include the string `attached` if the session is attached and the string `not attached` if it is unattached, or `#{?automatic-rename,yes,no}` will include `yes` if **`automatic-rename`** is enabled, or '`no`' if not. Conditionals can be nested arbitrarily. Inside a conditional, '`,`' and '`}`' must be escaped as '`#,`' and '`#}`', unless they are part of a `#{...}` replacement. For example:

```
#{?pane_in_mode,#[fg=white#,bg=red],#[fg=red#,bg=white]}#W .
```

String comparisons may be expressed by prefixing two comma-separated alternatives by '`==`', '`!=`', '`<`', '`>`', '`<=`' or '`>=`' and a colon. For example `#{==:#{host},myhost}` will be replaced by '`1`' if running on `myhost`, otherwise by '`0`'. '`||`' and '`&&`' evaluate to true if either or both of two comma-separated alternatives are true, for example `#{||:#{pane_in_mode},#{alternate_on}}`.

An '`m`' specifies an*fnmatc h*(3) or regular expression comparison. The first argument is the pattern and the second the string to compare. An optional argument specifies flags: '`r`' means the pattern is a regular expression instead of the default *fnmatch*(3) pattern, and '`i`' means to ignore case. For example: `#{m:*foo*,#{host}}` or `#{m/ri:^A,MYVAR}`. A '`C`' performs a search for an*fnmatc h*(3) pattern or regular expression in the pane content and evaluates to zero if not found, or a line number if found. Like '`m`', an '`r`' flag means search for a regular expression and '`i`' ignores case. For example: `#{C/r:^Start}`

Numeric operators may be performed by prefixing two comma-separated alternatives with an '`e`' and an operator. An optional '`f`' flag may be given after the operator to use floating point numbers, otherwise integers are used. This may be followed by a number giving the number of decimal places to use for the result. The available operators are: addition '`+`', subtraction '`-`', multiplication '`*`', division '`/`', modulus '`m`' or '`%`' (note that '`%`' must be escaped as '`%%`' in formats which are also expanded by *strftime*(3)) and numeric comparison operators '`==`', '`!=`', '`<`', '`<=`', '`>`' and '`>=`'. For example, `#{e|*|f|4:5.5,3}` multiplies 5.5 by 3 for a result with four decimal places and `#{e|%%:7,3}` returns the modulus of 7 and 3. '`a`' replaces a numeric argument by its ASCII equivalent, so `#{a:98}` results in '`b`'. '`c`' replaces a**`tmux`** colour by its six-digit hexadecimal RGB value.

A limit may be placed on the length of the resultant string by prefixing it by an '`=`', a number and a colon. Positive numbers count from the start of the string and negative from the end, so `#{=5:pane_title}` will include at most the first five characters of the pane title, or `#{=-5:pane_title}` the last five

characters. A suffix or prefix may be given as a second argument - if provided then it is appended or prepended to the string if the length has been trimmed, for example `#{=/5/...:pane_title}` will append `...` if the pane title is more than five characters. Similarly, '`p`' pads the string to a given width, for example `#{p10:pane_title}` will result in a width of at least 10 characters. A positive width pads on the left, a negative on the right. '`n`' expands to the length of the variable and '`w`' to its width when displayed, for example `#{n:window_name}`.

Prefixing a time variable with '`t:`' will convert it to a string, so if `#{window_activity}` gives `1445765102`, `#{t:window_activity}` gives `Sun Oct 25 09:25:02 2015`. Adding '`p (` `t/p`') will use shorter but less accurate time format for times in the past. A custom format may be given using an '`f`' suffix (note that '`%`' must be escaped as '`%%`' if the format is separately being passed through *strftime*(3), for example in the **status-left** option): `#{t/f/%%H#:%%M:window_activity}`, see *strftime*(3).

The '`b:`' and '`d:`' prefixes are *basename*(3) and *dirname*(3) of the variable respectively. '`q:`' will escape *sh*(1) special characters or with a '`h`' suffix, escape hash characters (so '`#`' becomes '`##`'). '`E:`' will expand the format twice, for example `#{E:status-left}` is the result of expanding the content of the **status-left** option rather than the option itself. '`T:`' is like '`E:`' but also expands *strftime*(3) specifiers. '`S:`', '`W:`', '`P:`' or '`L:`' will loop over each session, window, pane or client and insert the format once for each. For windows and panes, two comma-separated formats may be given: the second is used for the current window or active pane. For example, to get a list of windows formatted like the status line:

```
#{W:#{E:window-status-format} ,#{E:window-status-current-format} }
```

'`N:`' checks if a window (without any suffix or with the '`w`' suffix) or a session (with the '`s`' suffix) name exists, for example `'N/w:foo'` is replaced with 1 if a window named `foo` exists.

A prefix of the form `s/foo/bar/:` will substitute `foo` with `bar` throughout. The first argument may be an extended regular expression and a final argument may be '`i`' to ignore case, for example `s/a(.)/\1x/i:` would change `abABab` into `bxBxbx`. A different delimiter character may also be used, to avoid collisions with literal slashes in the pattern. For example, `s|foo/|bar/|:` will substitute `foo/` with `bar/` throughout.

In addition, the last line of a shell command's output may be inserted using `#()`. For example, `#(uptime)` will insert the system's uptime. When constructing formats, **tmux** does not wait for `#()` commands to finish; instead, the previous result from running the same command is used, or a placeholder if the command has not been run before. If the command hasn't exited, the most recent line of output will be used, but the status line will not be updated more than once a second. Commands are executed using */bin/sh* and with the **tmux** global environment set (see the "GLOBAL AND SESSION ENVIRONMENT" section).

An '`l`' specifies that a string should be interpreted literally and not expanded. For example `#{l:#{?pane_in_mode,yes,no}}` will be replaced by `#{?pane_in_mode,yes,no}`.

The following variables are available, where appropriate:

| Variable name | Alias | Replaced with |
|---|---|---|
| `active_window_index` | | Index of active window in session |
| `alternate_on` | | 1 if pane is in alternate screen |
| `alternate_saved_x` | | Saved cursor X in alternate screen |
| `alternate_saved_y` | | Saved cursor Y in alternate screen |
| `buffer_created` | | Time buffer created |
| `buffer_name` | | Name of buffer |
| `buffer_sample` | | Sample of start of buffer |
| `buffer_size` | | Size of the specified buffer in bytes |
| `client_activity` | | Time client last had activity |
| `client_cell_height` | | Height of each client cell in pixels |

| | | |
|---|---|---|
| `client_cell_width` | | Width of each client cell in pixels |
| `client_control_mode` | | 1 if client is in control mode |
| `client_created` | | Time client created |
| `client_discarded` | | Bytes discarded when client behind |
| `client_flags` | | List of client flags |
| `client_height` | | Height of client |
| `client_key_table` | | Current key table |
| `client_last_session` | | Name of the client’s last session |
| `client_name` | | Name of client |
| `client_pid` | | PID of client process |
| `client_prefix` | | 1 if prefix key has been pressed |
| `client_readonly` | | 1 if client is read-only |
| `client_session` | | Name of the client’s session |
| `client_termfeatures` | | Terminal features of client, if any |
| `client_termname` | | Terminal name of client |
| `client_termtype` | | Terminal type of client, if available |
| `client_tty` | | Pseudo terminal of client |
| `client_uid` | | UID of client process |
| `client_user` | | User of client process |
| `client_utf8` | | 1 if client supports UTF-8 |
| `client_width` | | Width of client |
| `client_written` | | Bytes written to client |
| `command` | | Name of command in use, if any |
| `command_list_alias` | | Command alias if listing commands |
| `command_list_name` | | Command name if listing commands |
| `command_list_usage` | | Command usage if listing commands |
| `config_files` | | List of configuration files loaded |
| `copy_cursor_hyperlink` | | Hyperlink under cursor in copy mode |
| `copy_cursor_line` | | Line the cursor is on in copy mode |
| `copy_cursor_word` | | Word under cursor in copy mode |
| `copy_cursor_x` | | Cursor X position in copy mode |
| `copy_cursor_y` | | Cursor Y position in copy mode |
| `current_file` | | Current configuration file |
| `cursor_character` | | Character at cursor in pane |
| `cursor_flag` | | Pane cursor flag |
| `cursor_x` | | Cursor X position in pane |
| `cursor_y` | | Cursor Y position in pane |
| `history_bytes` | | Number of bytes in window history |
| `history_limit` | | Maximum window history lines |
| `history_size` | | Size of history in lines |
| `hook` | | Name of running hook, if any |
| `hook_client` | | Name of client where hook was run, if any |
| `hook_pane` | | ID of pane where hook was run, if any |
| `hook_session` | | ID of session where hook was run, if any |
| `hook_session_name` | | Name of session where hook was run, if any |
| `hook_window` | | ID of window where hook was run, if any |
| `hook_window_name` | | Name of window where hook was run, if any |
| `host` | #H | Hostname of local host |
| `host_short` | #h | Hostname of local host (no domain name) |
| `insert_flag` | | Pane insert flag |
| `keypad_cursor_flag` | | Pane keypad cursor flag |
| `keypad_flag` | | Pane keypad flag |

| | | |
|---|---|---|
| `last_window_index` | | Index of last window in session |
| `line` | | Line number in the list |
| `mouse_all_flag` | | Pane mouse all flag |
| `mouse_any_flag` | | Pane mouse any flag |
| `mouse_button_flag` | | Pane mouse button flag |
| `mouse_hyperlink` | | Hyperlink under mouse, if any |
| `mouse_line` | | Line under mouse, if any |
| `mouse_sgr_flag` | | Pane mouse SGR flag |
| `mouse_standard_flag` | | Pane mouse standard flag |
| `mouse_status_line` | | Status line on which mouse event took place |
| `mouse_status_range` | | Range type or argument of mouse event on status line |
| `mouse_utf8_flag` | | Pane mouse UTF-8 flag |
| `mouse_word` | | Word under mouse, if any |
| `mouse_x` | | Mouse X position, if any |
| `mouse_y` | | Mouse Y position, if any |
| `next_session_id` | | Unique session ID for next new session |
| `origin_flag` | | Pane origin flag |
| `pane_active` | | 1 if active pane |
| `pane_at_bottom` | | 1 if pane is at the bottom of window |
| `pane_at_left` | | 1 if pane is at the left of window |
| `pane_at_right` | | 1 if pane is at the right of window |
| `pane_at_top` | | 1 if pane is at the top of window |
| `pane_bg` | | Pane background colour |
| `pane_bottom` | | Bottom of pane |
| `pane_current_command` | | Current command if available |
| `pane_current_path` | | Current path if available |
| `pane_dead` | | 1 if pane is dead |
| `pane_dead_signal` | | Exit signal of process in dead pane |
| `pane_dead_status` | | Exit status of process in dead pane |
| `pane_dead_time` | | Exit time of process in dead pane |
| `pane_fg` | | Pane foreground colour |
| `pane_format` | | 1 if format is for a pane |
| `pane_height` | | Height of pane |
| `pane_id` | #D | Unique pane ID |
| `pane_in_mode` | | 1 if pane is in a mode |
| `pane_index` | #P | Index of pane |
| `pane_input_off` | | 1 if input to pane is disabled |
| `pane_key_mode` | | Extended key reporting mode in this pane |
| `pane_last` | | 1 if last pane |
| `pane_left` | | Left of pane |
| `pane_marked` | | 1 if this is the marked pane |
| `pane_marked_set` | | 1 if a marked pane is set |
| `pane_mode` | | Name of pane mode, if any |
| `pane_path` | | Path of pane (can be set by application) |
| `pane_pid` | | PID of first process in pane |
| `pane_pipe` | | 1 if pane is being piped |
| `pane_right` | | Right of pane |
| `pane_search_string` | | Last search string in copy mode |
| `pane_start_command` | | Command pane started with |
| `pane_start_path` | | Path pane started with |
| `pane_synchronized` | | 1 if pane is synchronized |
| `pane_tabs` | | Pane tab positions |

| | | |
|---|---|---|
| `pane_title` | #T | Title of pane (can be set by application) |
| `pane_top` | | Top of pane |
| `pane_tty` | | Pseudo terminal of pane |
| `pane_unseen_changes` | | 1 if there were changes in pane while in mode |
| `pane_width` | | Width of pane |
| `pid` | | Server PID |
| `rectangle_toggle` | | 1 if rectangle selection is activated |
| `scroll_position` | | Scroll position in copy mode |
| `scroll_region_lower` | | Bottom of scroll region in pane |
| `scroll_region_upper` | | Top of scroll region in pane |
| `search_count` | | Count of search results |
| `search_count_partial` | | 1 if search count is partial count |
| `search_match` | | Search match if any |
| `search_present` | | 1 if search started in copy mode |
| `selection_active` | | 1 if selection started and changes with the cursor in copy mode |
| `selection_end_x` | | X position of the end of the selection |
| `selection_end_y` | | Y position of the end of the selection |
| `selection_present` | | 1 if selection started in copy mode |
| `selection_start_x` | | X position of the start of the selection |
| `selection_start_y` | | Y position of the start of the selection |
| `server_sessions` | | Number of sessions |
| `session_activity` | | Time of session last activity |
| `session_alerts` | | List of window indexes with alerts |
| `session_attached` | | Number of clients session is attached to |
| `session_attached_list` | | List of clients session is attached to |
| `session_created` | | Time session created |
| `session_format` | | 1 if format is for a session |
| `session_group` | | Name of session group |
| `session_group_attached` | | Number of clients sessions in group are attached to |
| `session_group_attached_list` | | List of clients sessions in group are attached to |
| `session_group_list` | | List of sessions in group |
| `session_group_many_attached` | | 1 if multiple clients attached to sessions in group |
| `session_group_size` | | Size of session group |
| `session_grouped` | | 1 if session in a group |
| `session_id` | | Unique session ID |
| `session_last_attached` | | Time session last attached |
| `session_many_attached` | | 1 if multiple clients attached |
| `session_marked` | | 1 if this session contains the marked pane |
| `session_name` | #S | Name of session |
| `session_path` | | Working directory of session |
| `session_stack` | | Window indexes in most recent order |
| `session_windows` | | Number of windows in session |
| `socket_path` | | Server socket path |
| `start_time` | | Server start time |
| `uid` | | Server UID |
| `user` | | Server user |
| `version` | | Server version |
| `window_active` | | 1 if window active |
| `window_active_clients` | | Number of clients viewing this window |
| `window_active_clients_list` | | List of clients viewing this window |
| `window_active_sessions` | | Number of sessions on which this window is active |

| | | |
|---|---|---|
| `window_active_sessions_list` | | List of sessions on which this window is active |
| `window_activity` | | Time of window last activity |
| `window_activity_flag` | | 1 if window has activity |
| `window_bell_flag` | | 1 if window has bell |
| `window_bigger` | | 1 if window is larger than client |
| `window_cell_height` | | Height of each cell in pixels |
| `window_cell_width` | | Width of each cell in pixels |
| `window_end_flag` | | 1 if window has the highest index |
| `window_flags` | #F | Window flags with # escaped as ## |
| `window_format` | | 1 if format is for a window |
| `window_height` | | Height of window |
| `window_id` | | Unique window ID |
| `window_index` | #I | Index of window |
| `window_last_flag` | | 1 if window is the last used |
| `window_layout` | | Window layout description, ignoring zoomed window panes |
| `window_linked` | | 1 if window is linked across sessions |
| `window_linked_sessions` | | Number of sessions this window is linked to |
| `window_linked_sessions_list` | | List of sessions this window is linked to |
| `window_marked_flag` | | 1 if window contains the marked pane |
| `window_name` | #W | Name of window |
| `window_offset_x` | | X offset into window if larger than client |
| `window_offset_y` | | Y offset into window if larger than client |
| `window_panes` | | Number of panes in window |
| `window_raw_flags` | | Window flags with nothing escaped |
| `window_silence_flag` | | 1 if window has silence alert |
| `window_stack_index` | | Index in session most recent stack |
| `window_start_flag` | | 1 if window has the lowest index |
| `window_visible_layout` | | Window layout description, respecting zoomed window panes |
| `window_width` | | Width of window |
| `window_zoomed_flag` | | 1 if window is zoomed |
| `wrap_flag` | | Pane wrap flag |

## STYLES

**`tmux`** offers various options to specify the colour and attributes of aspects of the interface, for example **`status-style`** for the status line. In addition, embedded styles may be specified in format options, such as **`status-left`**, by enclosing them in '`#[`' and '`]`'.

A style may be the single term `default` to specify the default style (which may come from an option, for example **`status-style`** in the status line) or a space or comma separated list of the following:

**`fg=colour`**
: Set the foreground colour. The colour is one of:**`black`**, **`red`**, **`green`**, **`yellow`**, **`blue`**, **`magenta`**, **`cyan`**, **`white`**; if supported the bright variants **`brightred`**, **`brightgreen`**, **`brightyellow`**; **`colour0`** to **`colour255`** from the 256-colour set; **`default`** for the default colour; **`terminal`** for the terminal default colour; or a hexadecimal RGB string such as `#ffffff`.

**`bg=colour`**
: Set the background colour.

**`us=colour`**
: Set the underscore colour.

**none** Set no attributes (turn off any active attributes).

**acs**, **bright** (or **bold**), **dim**, **underscore**, **blink**, **reverse**, **hidden**, **italics**, **overline**, **strikethrough**, **double-underscore**, **curly-underscore**, **dotted-underscore**, **dashed-underscore**
Set an attribute. Any of the attributes may be prefixed with 'no' to unset.**acs** is the terminal alternate character set.

**align=left** (or **noalign**), **align=centre**, **align=right**
Align text to the left, centre or right of the available space if appropriate.

**fill=colour**
Fill the available space with a background colour if appropriate.

**list=on**, **list=focus**, **list=left-marker**, **list=right-marker**, **nolist**
Mark the position of the various window list components in the **status-format** option: **list=on** marks the start of the list; **list=focus** is the part of the list that should be kept in focus if the entire list won't fit in the available space (typically the current window); **list=left-marker** and **list=right-marker** mark the text to be used to mark that text has been trimmed from the left or right of the list if there is not enough space.

**push-default**, **pop-default**
Store the current colours and attributes as the default or reset to the previous default. A **push-default** affects any subsequent use of the **default** term until a **pop-default**. Only one default may be pushed (each **push-default** replaces the previous saved default).

**range=left**, **range=right**, **range=session|X**, **range=window|X**, **range=pane|X**, **range=user|X**, **norange**
Mark a range for mouse events in the **status-format** option. When a mouse event occurs in the **range=left** or **range=right** range, the StatusLeft and StatusRight key bindings are triggered.

**range=session|X**, **range=window|X** and **range=pane|X** are ranges for a session, window or pane. These trigger the Status mouse key with the target session, window or pane given by the 'X' argument. 'X' is a session ID, window index in the current session or a pane ID. For these, the **mouse_status_range** format variable will be set to session, window or pane.

**range=user|X** is a user-defined range; it triggers the Status mouse key. The argument 'X' will be available in the **mouse_status_range** format variable. 'X' must be at most 15 bytes in length.

Examples are:

```
fg=yellow bold underscore blink
bg=black,fg=default,noreverse
```

## NAMES AND TITLES

**tmux** distinguishes between names and titles. Windows and sessions have names, which may be used to specify them in targets and are displayed in the status line and various lists: the name is the **tmux** identifier for a window or session. Only panes have titles. A pane's title is typically set by the program running inside the pane using an escape sequence (like it would set the *xterm*(1) window title in *X*(7)). Windows themselves do not have titles - a window's title is the title of its active pane. **tmux** itself may set the title of the terminal in which the client is running, see the **set-titles** option.

A session's name is set with the **new-session** and **rename-session** commands. A window's name is set with one of:

1. A command argument (such as −n for **new-window** or **new-session**).

2. An escape sequence (if the **allow-rename** option is turned on):

```
$ printf '\033kWINDOW_NAME\033\\'
```

3. Automatic renaming, which sets the name to the active command in the window's active pane. See the **automatic-rename** option.

When a pane is first created, its title is the hostname. A pane's title can be set via the title setting escape sequence, for example:

```
$ printf '\033]2;My Title\033\\'
```

It can also be modified with the **select-pane** `-T` command.

## GLOBAL AND SESSION ENVIRONMENT

When the server is started, **tmux** copies the environment into the *global environment*; in addition, each session has a *session environment*. When a window is created, the session and global environments are merged. If a variable exists in both, the value from the session environment is used. The result is the initial environment passed to the new process.

The **update-environment** session option may be used to update the session environment from the client when a new session is created or an old reattached. **tmux** also initialises the `TMUX` variable with some internal information to allow commands to be executed from inside, and the `TERM` variable with the correct terminal setting of `screen`.

Variables in both session and global environments may be marked as hidden. Hidden variables are not passed into the environment of new processes and instead can only be used by tmux itself (for example in formats, see the "FORMATS" section).

Commands to alter and view the environment are:

**set-environment** [`-Fhgru`] [`-t` *target-session*] *name* [*value*]
(alias: **setenv**)

Set or unset an environment variable. If `-g` is used, the change is made in the global environment; otherwise, it is applied to the session environment for *target-session*. If `-F` is present, then *value* is expanded as a format. The `-u` flag unsets a variable. `-r` indicates the variable is to be removed from the environment before starting a new process. `-h` marks the variable as hidden.

**show-environment** [`-hgs`] [`-t` *target-session*] [*variable*]
(alias: **showenv**)

Display the environment for *target-session* or the global environment with `-g`. If *variable* is omitted, all variables are shown. Variables removed from the environment are prefixed with '`-`'. If `-s` is used, the output is formatted as a set of Bourne shell commands. `-h` shows hidden variables (omitted by default).

## STATUS LINE

**tmux** includes an optional status line which is displayed in the bottom line of each terminal.

By default, the status line is enabled and one line in height (it may be disabled or made multiple lines with the **status** session option) and contains, from left-to-right: the name of the current session in square brackets; the window list; the title of the active pane in double quotes; and the time and date.

Each line of the status line is configured with the **status-format** option. The default is made of three parts: configurable left and right sections (which may contain dynamic content such as the time or output from a shell command, see the **status-left**, **status-left-length**, **status-right**, and **status-right-length** options below), and a central window list. By default, the window list shows the index, name and (if any) flag of the windows present in the current session in ascending numerical order. It may be customised with the *window-status-format* and *window-status-current-format* options. The flag is one of the following symbols appended to the window name:

| Symbol | Meaning |
|---|---|
| * | Denotes the current window. |
| - | Marks the last window (previously selected). |
| # | Window activity is monitored and activity has been detected. |
| ! | Window bells are monitored and a bell has occurred in the window. |
| ~ | The window has been silent for the monitor-silence interval. |
| M | The window contains the marked pane. |
| Z | The window's active pane is zoomed. |

The # symbol relates to the **monitor-activity** window option. The window name is printed in inverted colours if an alert (bell, activity or silence) is present.

The colour and attributes of the status line may be configured, the entire status line using the **status-style** session option and individual windows using the **window-status-style** window option.

The status line is automatically refreshed at interval if it has changed, the interval may be controlled with the **status-interval** session option.

Commands related to the status line are as follows:

**clear-prompt-history** [−T *prompt-type*]
(alias: **clearphist**)
Clear status prompt history for prompt type *prompt-type*. If −T is omitted, then clear history for all types. See **command-prompt** for possible values for *prompt-type*.

**command-prompt** [−1bFikN] [−I *inputs*] [−p *prompts*] [−t *target-client*] [−T *prompt-type*] [*template*]
Open the command prompt in a client. This may be used from inside **tmux** to execute commands interactively.

If *template* is specified, it is used as the command. With −F, *template* is expanded as a format.

If present, −I is a comma-separated list of the initial text for each prompt. If −p is given, *prompts* is a comma-separated list of prompts which are displayed in order; otherwise a single prompt is displayed, constructed from *template* if it is present, or ':' if not.

Before the command is executed, the first occurrence of the string '%%' and all occurrences of '%1' are replaced by the response to the first prompt, all '%2' are replaced with the response to the second prompt, and so on for further prompts. Up to nine prompt responses may be replaced ('%1' to '%9'). %%% is like '%%' but any quotation marks are escaped.

−1 makes the prompt only accept one key press, in this case the resulting input is a single character. −k is like −1 but the key press is translated to a key name. −N makes the prompt only accept numeric key presses. −i executes the command every time the prompt input changes instead of when the user exits the command prompt.

−T tells **tmux** the prompt type. This affects what completions are offered when *Tab* is pressed. Available types are: command, search, target and window-target.

The following keys have a special meaning in the command prompt, depending on the value of the **status-keys** option:

| Function | vi | emacs |
|---|---|---|
| Cancel command prompt | q | Escape |
| Delete from cursor to start of word | | C-w |
| Delete entire command | d | C-u |
| Delete from cursor to end | D | C-k |

| | | |
|---|---|---|
| Execute command | Enter | Enter |
| Get next command from history | | Do wn |
| Get previous command from history | | Up |
| Insert top paste buffer | p | C-y |
| Look for completions | T ab | Tab |
| Move cursor left | h | Left |
| Move cursor right | l | Right |
| Move cursor to end | $ | C-e |
| Move cursor to next word | w | M-f |
| Move cursor to previous word | b | M-b |
| Move cursor to start | 0 | C-a |
| Transpose characters | | C-t |

With −b, the prompt is shown in the background and the invoking client does not exit until it is dismissed.

**confirm-before** [−by] [−c *confirm-key*] [−p *prompt*] [−t *target-client*] *command*
(alias: **confirm**)

Ask for confirmation before executing *command*. If −p is gi ven, *prompt* is the prompt to display; otherwise a prompt is constructed from *command*. It may contain the special character sequences supported by the **status-left** option. With −b, the prompt is shown in the background and the invoking client does not exit until it is dismissed. −y changes the default behaviour (if Enter alone is pressed) of the prompt to run the command. −c changes the confirmation key to *confirm-key*; the default is ‘y’.

**display-menu** [−OM] [−b *border-lines*] [−c *target-client*] [−C *starting-choice*] [−H *selected-style*] [−s *style*] [−S *border-style*] [−t *target-pane*] [−T *title*] [−x *position*] [−y *position*] *name key command* [*argument ...*]
(alias: **menu**)

Display a menu on *target-client*. *target-pane* gives the target for any commands run from the menu.

A menu is passed as a series of arguments: first the menu item name, second the key shortcut (or empty for none) and third the command to run when the menu item is chosen. The name and command are formats, see the “FORMATS” and “STYLES” sections. If the name begins with a hyphen (-), then the item is disabled (shown dim) and may not be chosen. The name may be empty for a separator line, in which case both the key and command should be omitted.

−b sets the type of characters used for drawing menu borders. See **popup-border-lines** for possible values for *border-lines*.

−H sets the style for the selected menu item (see “STYLES”).

−s sets the style for the menu and −S sets the style for the menu border (see “STYLES”).

−T is a format for the menu title (see “FORMATS”).

−C sets the menu item selected by default, if the menu is not bound to a mouse key binding.

−x and −y give the position of the menu. Both may be a row or column number, or one of the following special values:

| Value | Flag | Meaning |
|---|---|---|
| C | Both | The centre of the terminal |
| R | −x | The right side of the terminal |
| P | Both | The bottom left of the pane |
| M | Both | The mouse position |
| W | Both | The window position on the status line |

| | | |
|---|---|---|
| S | −y | The line above or below the status line |

Or a format, which is expanded including the following additional variables:

| **Variable name** | **Replaced with** |
|---|---|
| popup_centre_x | Centered in the client |
| popup_centre_y | Centered in the client |
| popup_height | Height of menu or popup |
| popup_mouse_bottom | Bottom of at the mouse |
| popup_mouse_centre_x | Horizontal centre at the mouse |
| popup_mouse_centre_y | Vertical centre at the mouse |
| popup_mouse_top | Top at the mouse |
| popup_mouse_x | Mouse X position |
| popup_mouse_y | Mouse Y position |
| popup_pane_bottom | Bottom of the pane |
| popup_pane_left | Left of the pane |
| popup_pane_right | Right of the pane |
| popup_pane_top | Top of the pane |
| popup_status_line_y | Above or below the status line |
| popup_width | Width of menu or popup |
| popup_window_status_line_x | At the window position in status line |
| popup_window_status_line_y | At the status line showing the window |

Each menu consists of items followed by a key shortcut shown in brackets. If the menu is too large to fit on the terminal, it is not displayed. Pressing the key shortcut chooses the corresponding item. If the mouse is enabled and the menu is opened from a mouse key binding, releasing the mouse button with an item selected chooses that item and releasing the mouse button without an item selected closes the menu. −O changes this behaviour so that the menu does not close when the mouse button is released without an item selected the menu is not closed and a mouse button must be clicked to choose an item.

−M tells **tmux** the menu should handle mouse events; by default only menus opened from mouse key bindings do so.

The following keys are available in menus:

| **Key** | **Function** |
|---|---|
| Enter | Choose selected item |
| Up | Select previous item |
| Down | Select next item |
| q | Exit menu |

**display-message** [−aIlNpv] [−c *target-client*] [−d *delay*] [−t *target-pane*] [*message*]

(alias: **display**)

Display a message. If −p is given, the output is printed to stdout, otherwise it is displayed in the *target-client* status line for up to *delay* milliseconds. If *delay* is not given, the **display-time** option is used; a delay of zero waits for a key press. ‘N’ ignores key presses and closes only after the delay expires. If−l is given, *message* is printed unchanged. Otherwise, the format of *message* is described in the “FORMATS” section; information is taken from *target-pane* if −t is given, otherwise the active pane.

−v prints verbose logging as the format is parsed and −a lists the format variables and their values.

−I forwards any input read from stdin to the empty pane given by *target-pane*.

**display-popup** [−BCE] [−b *border-lines*] [−c *target-client*] [−d *start-directory*] [−e *environment*] [−h *height*] [−s *border-style*] [−S *style*] [−t *target-pane*] [−T *title*] [−w *width*] [−x *position*] [−y

*position*] [*shell-command*]
(alias: **popup**)
Display a popup running *shell-command* on *target-client*. A popup is a rectangular box drawn over the top of any panes. Panes are not updated while a popup is present.

-E closes the popup automatically when *shell-command* exits. Two -E closes the popup only if *shell-command* exited with success.

-x and -y give the position of the popup, they have the same meaning as for the **display-menu** command. -w and -h give the width and height - both may be a percentage (followed by '%'). If omitted, half of the terminal size is used.

-B does not surround the popup by a border.

-b sets the type of characters used for drawing popup borders. When -B is specified, the -b option is ignored. See **popup-border-lines** for possible values for *border-lines*.

-s sets the style for the popup and -S sets the style for the popup border (see "STYLES").

-e takes the form VARIABLE=value and sets an environment variable for the popup; it may be specified multiple times.

-T is a format for the popup title (see "FORMATS").

The -C flag closes any popup on the client.

**show-prompt-history** [-T *prompt-type*]
(alias: **showphist**)
Display status prompt history for prompt type *prompt-type*. If-T is omitted, then sho w history for all types. See **command-prompt** for possible values for *prompt-type*.

**BUFFERS**

**tmux** maintains a set of named *paste buffers*. Each buffer may be either explicitly or automatically named. Explicitly named buffers are named when created with the **set-buffer** or **load-buffer** commands, or by renaming an automatically named buffer with **set-buffer** -n. Automatically named buffers are given a name such as buffer0001, buffer0002 and so on. When the **buffer-limit** option is reached, the oldest automatically named buffer is deleted. Explicitly named buffers are not subject to **buffer-limit** and may be deleted with the **delete-buffer** command.

Buffers may be added using **copy-mode** or the **set-buffer** and **load-buffer** commands, and pasted into a window using the **paste-buffer** command. If a buffer command is used and no buffer is specified, the most recently added automatically named buffer is assumed.

A configurable history buffer is also maintained for each window. By default, up to 2000 lines are kept; this can be altered with the **history-limit** option (see the **set-option** command above).

The buffer commands are as follows:

**choose-buffer** [-NZr] [-F *format*] [-f *filter*] [-K *key-format*] [-O *sort-order*] [-t *target-pane*] [*template*]
Put a pane into buffer mode, where a buffer may be chosen interactively from a list. Each buffer is shown on one line. A shortcut key is shown on the left in brackets allowing for immediate choice, or the list may be navigated and an item chosen or otherwise manipulated using the keys below. -Z zooms the pane. The following keys may be used in buffer mode:

| Key | Function |
|---|---|
| Enter | Paste selected buffer |
| Up | Select previous buffer |
| Down | Select next buffer |
| C-s | Search by name or content |

| | |
|---|---|
| `n` | Repeat last search forwards |
| `N` | Repeat last search backwards |
| `t` | Toggle if buffer is tagged |
| `T` | Tag no buffers |
| `C-t` | Tag all buffers |
| `p` | Paste selected buffer |
| `P` | Paste tagged buffers |
| `d` | Delete selected buffer |
| `D` | Delete tagged buffers |
| `e` | Open the buffer in an editor |
| `f` | Enter a format to filter items |
| `O` | Change sort field |
| `r` | Reverse sort order |
| `v` | Toggle preview |
| `q` | Exit mode |

After a buffer is chosen, '`%%`' is replaced by the buffer name in *`template`* and the result executed as a command. If *`template`* is not given, "paste-buffer -p -b '%%'" is used.

`-O` specifies the initial sort field: one of `time` (creation), `name` or `size`. `-r` reverses the sort order. `-f` specifies an initial filter: the filter is a format - if it evaluates to zero, the item in the list is not shown, otherwise it is shown. If a filter would lead to an empty list, it is ignored. `-F` specifies the format for each item in the list and `-K` a format for each shortcut key; both are evaluated once for each line. `-N` starts without the preview. This command works only if at least one client is attached.

**`clear-history`** [`-H`] [`-t` *`target-pane`*]
(alias: **`clearhist`**)
Remove and free the history for the specified pane. `-H` also removes all hyperlinks.

**`delete-buffer`** [`-b` *`buffer-name`*]
(alias: **`deleteb`**)
Delete the buffer named *`buffer-name`*, or the most recently added automatically named buffer if not specified.

**`list-buffers`** [`-F` *`format`*] [`-f` *`filter`*]
(alias: **`lsb`**)
List the global buffers. `-F` specifies the format of each line and `-f` a filter. Only buffers for which the filter is true are shown. See the "FORMATS" section.

**`load-buffer`** [`-w`] [`-b` *`buffer-name`*] [`-t` *`target-client`*] *`path`*
(alias: **`loadb`**)
Load the contents of the specified paste buffer from *`path`*. If `-w` is given, the buffer is also sent to the clipboard for *`target-client`* using the *xterm*(1) escape sequence, if possible. If *`path`* is '`-`', the contents are read from stdin.

**`paste-buffer`** [`-dpr`] [`-b` *`buffer-name`*] [`-s` *`separator`*] [`-t` *`target-pane`*]
(alias: **`pasteb`**)
Insert the contents of a paste buffer into the specified pane. If not specified, paste into the current one. With `-d`, also delete the paste buffer. When output, any linefeed (LF) characters in the paste buffer are replaced with a separator, by default carriage return (CR). A custom separator may be specified using the `-s` flag. The `-r` flag means to do no replacement (equivalent to a separator of LF). If `-p` is specified, paste bracket control codes are inserted around the buffer if the application has requested bracketed paste mode.

**`save-buffer`** [`-a`] [`-b` *`buffer-name`*] *`path`*
(alias: **`saveb`**)
Save the contents of the specified paste buffer to *`path`*. The `-a` option appends to rather than overwriting the file. If *`path`* is '`-`', the contents are read from stdin.

**set-buffer** [−aw] [−b *buffer-name*] [−t *target-client*] [−n *new-buffer-name*] *data*
(alias: **setb**)
Set the contents of the specified buffer to *data*. If −w is given, the buffer is also sent to the clipboard for *target-client* using the *xterm*(1) escape sequence, if possible. The −a option appends to rather than overwriting the buffer. The −n option renames the buffer to *new-buffer-name*.

**show-buffer** [−b *buffer-name*]
(alias: **showb**)
Display the contents of the specified buffer.

## MISCELLANEOUS

Miscellaneous commands are as follows:

**clock-mode** [−t *target-pane*]
Display a large clock.

**if-shell** [−bF] [−t *target-pane*] *shell-command command* [*command*]
(alias: **if**)
Execute the first *command* if *shell-command* (run with */bin/sh*) returns success or the second *command* otherwise. Before being executed, *shell-command* is expanded using the rules specified in the "FORMATS" section, including those relevant to *target-pane*. With −b, *shell-command* is run in the background.

If −F is given, *shell-command* is not executed but considered success if neither empty nor zero (after formats are expanded).

**lock-server**
(alias: **lock**)
Lock each client individually by running the command specified by the **lock-command** option.

**run-shell** [−bC] [−c *start-directory*] [−d *delay*] [−t *target-pane*] [*shell-command*]
(alias: **run**)
Execute *shell-command* using */bin/sh* or (with −C) a **tmux** command in the background without creating a window. Before being executed, *shell-command* is expanded using the rules specified in the "FORMATS" section. With −b, the command is run in the background. −d waits for *delay* seconds before starting the command. If −c is given, the current working directory is set to *start-directory*. If −C is not given, any output to stdout is displayed in view mode (in the pane specified by −t or the current pane if omitted) after the command finishes. If the command fails, the exit status is also displayed.

**wait-for** [−L | −S | −U] *channel*
(alias: **wait**)
When used without options, prevents the client from exiting until woken using **wait-for** −S with the same channel. When −L is used, the channel is locked and any clients that try to lock the same channel are made to wait until the channel is unlocked with **wait-for** −U.

## EXIT MESSAGES

When a **tmux** client detaches, it prints a message. This may be one of:

detached (from session ...)
The client was detached normally.

detached and SIGHUP
The client was detached and its parent sent the SIGHUP signal (for example with **detach-client** −P).

lost tty The client's *tty*(4) or *pty*(4) was unexpectedly destroyed.

terminated
: The client was killed with `SIGTERM`.

too far behind
: The client is in control mode and became unable to keep up with the data from **tmux**.

exited The server exited when it had no sessions.

server exited
: The server exited when it received `SIGTERM`.

server exited unexpectedly
: The server crashed or otherwise exited without telling the client the reason.

## TERMINFO EXTENSIONS

**tmux** understands some unofficial extensions to *terminfo*(5). It is not normally necessary to set these manually, instead the **terminal-features** option should be used.

*AX* An existing extension that tells **tmux** the terminal supports default colours.

*Bidi* Tell **tmux** that the terminal supports the VTE bidirectional text extensions.

*Cs*, *Cr* Set the cursor colour. The first takes a single string argument and is used to set the colour; the second takes no arguments and restores the default cursor colour. If set, a sequence such as this may be used to change the cursor colour from inside **tmux**:

```
$ printf '\033]12;red\033\\'
```

The colour is an *X*(7) colour, see *XParseColor*(3).

*Cmg*, *Clmg*, *Dsmg*, *Enmg*
: Set, clear, disable or enable DECSLRM margins. These are set automatically if the terminal reports it is *VT420* compatible.

*Dsbp*, *Enbp*
: Disable and enable bracketed paste. These are set automatically if the *XT* capability is present.

*Dseks*, *Eneks*
: Disable and enable extended keys.

*Dsfcs*, *Enfcs*
: Disable and enable focus reporting. These are set automatically if the *XT* capability is present.

*Hls* Set or clear a hyperlink annotation.

*Nobr* Tell **tmux** that the terminal does not use bright colors for bold display.

*Rect* Tell **tmux** that the terminal supports rectangle operations.

*Smol* Enable the overline attribute.

*Smulx* Set a styled underscore. The single parameter is one of: 0 for no underscore, 1 for normal underscore, 2 for double underscore, 3 for curly underscore, 4 for dotted underscore and 5 for dashed underscore.

*Setulc*, *Setulc1*, *ol*
: Set the underscore colour or reset to the default. *Setulc* is for RGB colours and *Setulc1* for ANSI or 256 colours. The *Setulc* argument is (red ∗ 65536) + (green ∗ 256) + blue where each is between 0 and 255.

*Ss*, *Se* Set or reset the cursor style. If set, a sequence such as this may be used to change the cursor to an underline:

```
$ printf '\033[4 q'
```

If *Se* is not set, Ss with argument 0 will be used to reset the cursor style instead.

*Swd* Set the opening sequence for the working directory notification. The sequence is terminated using the standard *fsl* capability.

*Sxl* Indicates that the terminal supports SIXEL.

*Sync* Start (parameter is 1) or end (parameter is 2) a synchronized update.

*Tc* Indicate that the terminal supports the `direct colour` RGB escape sequence (for example, \e[38;2;255;255;255m).

If supported, this is used for the initialize colour escape sequence (which may be enabled by adding the `initc` and `ccc` capabilities to the **tmux** *terminfo*(5) entry).

This is equivalent to the *RGB terminfo*(5) capability.

*Ms* Store the current buffer in the host terminal's selection (clipboard). See the *set-clipboard* option above and the *xterm*(1) man page.

*XT* This is an existing extension capability that tmux uses to mean that the terminal supports the *xterm*(1) title set sequences and to automatically set some of the capabilities above.

## CONTROL MODE

**tmux** offers a textual interface called *control mode*. This allows applications to communicate with **tmux** using a simple text-only protocol.

In control mode, a client sends **tmux** commands or command sequences terminated by newlines on standard input. Each command will produce one block of output on standard output. An output block consists of a *%begin* line followed by the output (which may be empty). The output block ends with a *%end* or *%error*. *%begin* and matching *%end* or *%error* have three arguments: an integer time (as seconds from epoch), command number and flags (currently not used). For example:

```
%begin 1363006971 2 1
0: ksh* (1 panes) [80x24] [layout b25f,80x24,0,0,2] @2 (active)
%end 1363006971 2 1
```

The **refresh-client** `-C` command may be used to set the size of a client in control mode.

In control mode, **tmux** outputs notifications. A notification will never occur inside an output block.

The following notifications are defined:

**%client-detached** *client*
The client has detached.

**%client-session-changed** *client session-id name*
The client is now attached to the session with ID *session-id*, which is named *name*.

**%config-error** *error*
An error has happened in a configuration file.

**%continue** *pane-id*
The pane has been continued after being paused (if the *pause-after* flag is set, see **refresh-client** `-A`).

**%exit** [*reason*]
The **tmux** client is exiting immediately, either because it is not attached to any session or an error occurred. If present,*reason* describes wh y the client exited.

**%extended-output** *pane-id age ... : value*
New form of **%output** sent when the *pause-after* flag is set. *age* is the time in milliseconds for which tmux had buffered the output before it was sent. Any subsequent arguments up until a single '`:`' are for future use and should be ignored.

**%layout-change** *window-id window-layout window-visible-layout window-flags*
The layout of a window with ID *window-id* changed. The new layout is *window-layout*. The window's visible layout is *window-visible-layout* and the window flags are *window-flags*.

**%message** *message*
A message sent with the **display-message** command.

**%output** *pane-id value*
A window pane produced output. *value* escapes non-printable characters and backslash as octal \xxx.

**%pane-mode-changed** *pane-id*
The pane with ID *pane-id* has changed mode.

**%paste-buffer-changed** *name*
Paste buffer *name* has been changed.

**%paste-buffer-deleted** *name*
Paste buffer *name* has been deleted.

**%pause** *pane-id*
The pane has been paused (if the *pause-after* flag is set).

**%session-changed** *session-id name*
The client is now attached to the session with ID *session-id*, which is named *name*.

**%session-renamed** *name*
The current session was renamed to *name*.

**%session-window-changed** *session-id window-id*
The session with ID *session-id* changed its active window to the window with ID *window-id*.

**%sessions-changed**
A session was created or destroyed.

**%subscription-changed** *name session-id window-id window-index pane-id ...* : *value*
The value of the format associated with subscription *name* has changed to *value*. See **refresh-client** **−B**. Any arguments after *pane-id* up until a single '**:**' are for future use and should be ignored.

**%unlinked-window-add** *window-id*
The window with ID *window-id* was created but is not linked to the current session.

**%unlinked-window-close** *window-id*
The window with ID *window-id*, which is not linked to the current session, was closed.

**%unlinked-window-renamed** *window-id*
The window with ID *window-id*, which is not linked to the current session, was renamed.

**%window-add** *window-id*
The window with ID *window-id* was linked to the current session.

**%window-close** *window-id*
The window with ID *window-id* closed.

**%window-pane-changed** *window-id pane-id*
The active pane in the window with ID *window-id* changed to the pane with ID *pane-id*.

**%window-renamed** *window-id name*
The window with ID *window-id* was renamed to *name*.

## ENVIRONMENT

When **tmux** is started, it inspects the following environment variables:

EDITOR If the command specified in this variable contains the string 'vi' andVISUAL is unset, use vi-style key bindings. Overridden by the **mode-keys** and **status-keys** options.

HOME The user's login directory. If unset, the*passwd*(5) database is consulted.

LC_CTYPE The character encoding *locale*(1). It is used for two separate purposes. For output to the terminal, UTF-8 is used if the -u option is gi ven or if LC_CTYPE contains "UTF-8" or "UTF8". Otherwise, only ASCII characters are written and non-ASCII characters are replaced with underscores ('_'). For input, **tmux** always runs with a UTF-8 locale. If en_US.UTF-8 is provided by the operating system, it is used and LC_CTYPE is ignored for input. Otherwise, LC_CTYPE tells **tmux** what the UTF-8 locale is called on the current system. If the locale specified by LC_CTYPE is not available or is not a UTF-8 locale, **tmux** exits with an error message.

LC_TIME The date and time format *locale*(1). It is used for locale-dependent*strftime*(3) format specifiers.

PWD The current working directory to be set in the global environment. This may be useful if it contains symbolic links. If the value of the variable does not match the current working directory, the variable is ignored and the result of *getcwd*(3) is used instead.

SHELL The absolute path to the default shell for new windows. See the**default-shell** option for details.

TMUX_TMPDIR
The parent directory of the directory containing the server sockets. See the-L option for details.

VISUAL If the command specified in this variable contains the string 'vi', use vi-style key bindings. Overridden by the **mode-keys** and **status-keys** options.

## FILES

*~/.tmux.conf*
*$XDG_CONFIG_HOME/tmux/tmux.conf*
*~/.config/tmux/tmux.conf* Default **tmux** configuration file.
*/etc/tmux.conf* System-wide configuration file.

## EXAMPLES

To create a new **tmux** session running *vi*(1):

```
$ tmux new-session vi
```

Most commands have a shorter form, known as an alias. For new-session, this is **new**:

```
$ tmux new vi
```

Alternatively, the shortest unambiguous form of a command is accepted. If there are several options, they are listed:

```
$ tmux n
ambiguous command: n, could be: new-session, new-window, next-window
```

Within an active session, a new window may be created by typing C-b c (Ctrl followed by the 'b' key followed by the 'c' key).

Windows may be navigated with: C-b 0 (to select window 0), C-b 1 (to select window 1), and so on; C-b nto select the ne xt window; and C-b p to select the previous window.

A session may be detached using `C-b d` (or by an external event such as *ssh*(1) disconnection) and reattached with:

```
$ tmux attach-session
```

Typing `C-b ?` lists the current key bindings in the current window; up and down may be used to navigate the list or ‘`q`’ to exit from it.

Commands to be run when the **`tmux`** server is started may be placed in the *~/.tmux.conf* configuration file. Common examples include:

Changing the default prefix key:

```
set-option -g prefix C-a
unbind-key C-b
bind-key C-a send-prefix
```

Turning the status line off, or changing its colour:

```
set-option -g status off
set-option -g status-style bg=blue
```

Setting other options, such as the default command, or locking after 30 minutes of inactivity:

```
set-option -g default-command "exec /bin/ksh"
set-option -g lock-after-time 1800
```

Creating new key bindings:

```
bind-key b set-option status
bind-key / command-prompt "split-window 'exec man %%'"
bind-key S command-prompt "new-window -n %1 'ssh %1'"
```

## SEE ALSO

*pty*(4)

## AUTHORS

Nicholas Marriott <*nicholas.marriott@gmail.com*>