The tmux terminal multiplexer

Nicholas Marriott, 2011

tmux is a program which allows multiple text terminal (TTY) processes to be created and managed from a single text terminal. The set of processes running in tmux may be detached from the terminal, continue running in the background and reattached to a different terminal without interruption. It provides the ability to display output on disparate terminals simultaneously and a large set of features for management of child terminals.

In development since 2007, from the outset tmux was primarily developed on OpenBSD and since mid-2009 has been part of the OpenBSD base system.

This paper presents a brief summary of the motivation for and history of the tmux program, an overview of the tmux design and main data structures, a discussion of the design decisions behind some of the components, and a summary of major limitations and areas for improvement.

History

tmux was started in 2007 with the aim of producing an alternative to the GNU screen program with some fundamental design differences.

GNU screen is a “full-screen window manager”[1] under sporadic development since the 1980s[2].

tmux was created with the following differences in mind:

- For a server-client design with all windows managed by a single server.
- To be fully BSD-licensed.
- To aim for a consistent, modern code
- style that would be easy to extend. In addition, the following are major goals:
- Try to have sensible defaults.
- Be conservative and correct: fix bugs quickly, fail fast on errors and limit incorrect output to terminals.
- Provide a full command interface and prefer the shell for scripting.
- Strive for a consistent interface: command syntax should be identical whether used from the shell or a key binding.
- Try to be well documented in manpages.
- Attempt to be portable but minimise dependencies outside the OpenBSD base system.

tmux and OpenBSD

tmux was imported into the OpenBSD base system in July 2009, a few months before the 4.6 release, to replace the window(1) program and is now installed as /usr/bin/tmux.

From the outset, tmux was written with OpenBSD code style and practices in mind and with OpenBSD as the primary development platform, so import into OpenBSD required relatively few changes, mainly to strip out portability code and add an OpenBSD-style Makefile.

Since then, tmux has changed to reuse existing OpenBSD code where appropriate, notably libevent and the imsg framework discussed in later sections. OpenBSD principles that tmux attempts to follow include:

- Use imsg for IPC.
- Maintain a consistent code style (largely following style(9)).
- Strive for code correctness; use safe functions such as strlcpy(3), strlcat(3) and strtonum(3).
- Use the ISC license.
- Attempt to provide sensible defaults; avoid

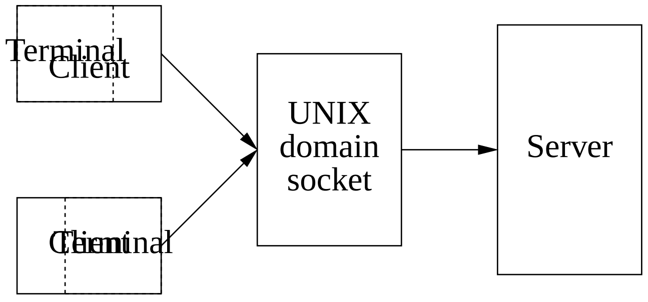
arbitrary changes to defaults without good reason.
- Undertake code inspection and auditing both manually and with automated tools such as lint.

Import into OpenBSD has brought tmux to the attention of a large and technically-proficient user and developer base as well as allowing its developer to work on a highly enjoyable and interesting project beyond tmux itself.

Design Overview

This section gives a description of the tmux design in terms of the primary data structures and the event loop.

tmux is a server-client system. All child processes, pseudoterminals, configuration and data structures are managed by one server process. The tmux server process may be attached to by one or more client processes. Clients have two primary roles: firstly, to transfer the environment and terminal file descriptors to the server; secondly, to occupy the terminal and prevent other processes from using it until the client exits (that is, it is detached). Clients attach to a server through a Unix domain socket and uses libevent[3], the OpenBSD imsg framework and a simple binary protocol to communicate.



Terminal
Client
UNIX domain socket
Server
Client
Terminal