Extending BarnOwl

Nelson Elhage

Student Information Processing Board

January 12, 2009

Nelson Elhage (SIPB)

Extending BarnOwl

э. January 12, 2009 1 / 30

< 4 ₽ × <

э

BarnOwl

- BarnOwl is a console IM client (not just Zephyr!)
- It's (secretly) extremely extensible and customizable
- It's works out of the box, but you can tweak it to your whims.

3

(日) (同) (三) (三)

Two ways to customize

- BarnOwl's built-in command "language" and tools
- Perl extensions

(日) (周) (三) (三)

3

Places to put code

~/.owl/startup List of BarnOwl commands executed at startup

The :startup command adds code to this file
~/.barnowlconf Perl code loaded at startup.
~/.owl/modules BarnOwl Modules – more about this later.

(日) (周) (三) (三)

BarnOwl's command "language"

- BarnOwl's functionality is exposed as a set of "commands"
- These commands are generally self-documented
- You can call commands interactively with ':' or 'M-x', or programmatically in various ways.
- :show commands lists all commands
- :show command command documents a command

Aliases

- The easiest way to define new commands is with :alias.
- :alias command expansion
- Like shell aliases, arguments to the *command* are appended to *expansion*
- :alias q error "I'm sorry, Dave, I can't let you do that"
- :alias v exec remctl zsr.mit.edu volume

(日) (周) (三) (三)

Filters

- You can define *filters* to match some subset of messages.
- These are most commonly used for coloring or limiting your view.

:filter creates a named filter.

:view restricts to a named filter.

< ⊢□

Filters – Syntax

- Basic syntax is *field regex*
- POSIX regexes see man 7 regex
- and, or, not
- Parentheses work but require spaces
- Filters can reference other filters using filter filter-name

3

(日) (同) (三) (三)

Filters – Examples

- class ^nelhage\$
- filter zephyr and class ^(un)*nelhage\$
- filter zephyr and filter personal and
 (sender ^geofft\$ or recipient ^geofft\$)
- filter user-geofft or filter user-broder

(日) (周) (三) (三)

What fields are available?

• Learn from BarnOwl's autogenerated filters

- smartnarrow ('M-n' and friends) automatically generate filters.
- :show filters will list all filters
- :show filter *filter* to show a specific one

• Press 'i' (:show info) and look at the "Owl Message Attributes"

-> sl			ORTNOY
	\$2		
	Owl Message Attri	ibutes:	
	type	: zephyr	
	sender	: sly@ATHENA.MIT.EDU	
	class	: sipb	
		: office	
	recipient		
	opcode		
	zsig	: Devil in the Shape of a Woman	
	realm	: ATHENA.MIT.EDU	
	body	: you can tell it's IAP =)	
	End (Press 'd	q' to quit)	
(1			

< ロト < 同ト < ヨト < ヨト

Special Filters

BarnOwl uses certain named filters for special purposes.

trash Defines "trash" messages. :delete trash (bound to 'T')
will automatically mark all of these as deleted.

reply-lockout :reply ('r' and friends) will not reply to any message matching this filter. Useful for lurking.

personal Defines what a "personal" message is. Don't change this unless you know what you're doing.

イロト イポト イヨト イヨト 二日

 BarnOwl's key-bindings are completely customizable.
 BarnOwl defines 7 built-in hierarchical keymaps: global System-wide default key bindings edit Text editing and command window editmulti Multi-line text editing editline Single-line text editing editresponse Single-line response to question popless Pop-up window (e.g. :show help) recv Main window /message list

・ロト ・ 同ト ・ ヨト ・ ヨト

Manipulating Keymaps

Commands

- show keymaps will show all keymaps and bindings
- :show keymap keymap will show a specific keymap
- :bindkey keymap key command command to rebind a key sequence.
- (Yes, that's a literal "command")

Examples

- :bindkey recv s command pperl BarnOwl::getcurmsg()->zsig
- :bindkey recv E command show errors
- :bindkey edit C-w command edit:delete-prev-word

BarnOwl Variables

- BarnOwl has its own system of configuration variables
- Use :getvar and :set to manipulate variables
- :show variables and :show variable *variable* for documentation.

- 32

(日) (周) (三) (三)

Variables

Some Useful Variables

appendtosepbar Add text to the bar between the edit window and the message list. colorztext Enable/Disable color. zsender Zephyr spoofing made easy! zsigproc Path to a program to generate zsigs.

- 3

(日) (周) (三) (三)

Setting up logging

From a shell:

- \$ mkdir -p Private/zlog/personal
- \$ mkdir -p Private/zlog/class

In BarnOwl:
:set logpath ~/Private/zlog/personal
:set classlogpath ~/Private/zlog/class
:set logging on
:set classlogging on

BarnOwl Modules

- BarnOwl has an extensible module system
 - You can pick-and-choose perl plugins to import into your BarnOwl
- There are two supported module formats

PAR Perl PAR files are essentially a zip'd directory tree bare An unpacked directory tree containing perl files.

- Modules can be installed locally or site-wide
 - Local modules go in ~/.owl/modules

Using Modules

- (We'll talk about creating them later)
- Copy the .par or directory tree into ~/.owl/modules.
 - The filename is important keep it the same.
- To reload the module into a running BarnOwl:
 - :reload-module *module* to reload one module
 - :reload-modules to reload all modules
 - As the names suggest, these can also be used to update modules to a newer version without restarting.

イロト 不得下 イヨト イヨト

Some modules we've developed...

ColorUtils asedeno's module to make coloring classes easy.

VT_ASedeno asedeno's VT-like style. I have my own variant of it.

- Alias Lets you change the displayed name of classes. Good for secret classes or ones with long names.
- ZStatus Send those silly Zephyr progress bars you may have seen all over Zephyr a while back
- DevUtils Provides a :eperl command to help developing perl in BarnOwl

Except for ColorUtils, these all live in my Public (/mit/nelhage/Public/BarnOwl).

(日) (周) (三) (三)

Coloring Zephyrs

Install ColorUtils from

~asedeno/BarnOwl/barnowl-native-modules

- :setcolor *color* sets the color for messages "like" the current message (e.g. same class, or same user for personals)
- :savecolors and :loadcolors save and load your colors to disk.
- 'c' is bound to start a :setcolor command.
- (And all these commands are documented in BarnOwl!)

IRC in your BarnOwl!

- There is (somewhat minimal) IRC support for BarnOwl!
- The IRC module in my Public (and in the source tree)
- Set irc:nick to your preferred nickname
- :irc-connect, :irc-join, :irc-msg...
- Commands are designed to be similar to traditional IRC clients, but with irc- instead of /
- :bindkey recv / start-command irc-

イロト 不得下 イヨト イヨト

Twitter in your BarnOwl!

- I hacked together a Twitter module for BarnOwl.
- This one isn't in the source tree git module in my Public
- Send and receive Twitters (and Twitter direct messages)
- Set twitter:class, twitter:instance and twitter:opcode to mirror selected Zephyrs to Twitter!

Module Format

- A module is a perl module in the BarnOwl::Module::Module::ModuleName package.
- And lives in ModuleDir/ModuleName (Or ModuleName.par)
- So a minimal module structure is:

[nelhage@phanatique:~/.owl/modules]\$ tree DemoModule
DemoModule

3 directories, 1 file

イロト 不得下 イヨト イヨト

BarnOwl commands from Perl

- The most basic thing Perl can do is execute any BarnOwl command.
- BarnOwl::command(command, args...)
 - With one argument, it gets tokenized like : would.
 - With more than one argument, they're passed as the command's argument list.
- BarnOwl::foo(args) is autoloaded to call the foo command.
 - (Currently) the arguments are always joined with spaces and re-tokenized.
- perlwrap.pm in the source tree documents essentially everything else.

イロト 不得下 イヨト イヨト 二日

Module hook points

- There are several main things a module might do:
 - Register *hooks* to take action when some event happens.
 - Define new commands and variables.
 - Insert messages into the message list.
 - Define a new message style.

(日) (同) (三) (三)

Taking action on events

- BarnOwl::Hook defines a class to register watchers for some event.
- Mostly you care about \$hook->add(FUNCTION)
 - FUNCTION can be either a subref or the name of a function.

()

Available hooks

These are all variables in the BarnOwl::Hooks module.
 \$startup Called on startup and module reload
 \$shutdown Called before shutdown
 \$receiveMessage Called when a message is received
 \$newMessage Called when any new message is added.
 \$mainLoop Called at least once/sec.
 \$getBuddyList Called to generate the buddy list.
 \$getQuickstart Called to generate :help quickstart.

イロト 不得下 イヨト イヨト

Idempotency

- Code in \$startup is currently called whenever *any* module is reloaded.
- Code at toplevel is called whenever this module is reloaded
- So it's important that code in both locations be idempotent.

12 N 4 12 N

< 4 → <

Some idioms

Instead of	Use
my \$var = F00;	our \$var;
	\$var = FOO unless
	<pre>defined(\$var);</pre>
<pre>\$hook->add(\&mySub);</pre>	\$hook->add(
	"BarnOwl::Module::"
	"MyModule::mySub");
&mySub	<pre>sub{mySub()}</pre>

Ξ.

・ロト ・四ト ・ヨト ・ヨト

A tour of some source...

- I'll demonstrate some more of the API by doing some source-diving.
- If you want to follow along, the source is at http://github.com/nelhage/barnowl/

∃ → (∃ →

< A[™] →

Contacting Us

- The source is at http://github.com/nelhage/barnowl/
- Email barnowl-dev@mit.edu
- Zephyr -c barnowl

- 31