Степан Заступов

Завидный конкурент bash

Наверное, каждый пользователь под понятием shell подразумевает именно bash. Не спорю, эта командная оболочка действительно не такая плохая, иначе ее не стали бы использовать по умолчанию в большинстве Linux-дистрибутивов. Однако, будь она действительно идеальна, не существовало бы альтернативных программных продуктов. Я не говорю про csh/tcsh, так как у него совершенно другой путь развития, и он уже стал привычной оболочкой для многих BSD-администраторов. Но зачем использовать то, что установлено по умолчанию, если есть альтернатива? Имя ей — zsh!

Zsh — пожалуй, самая «навороченная» и динамично развивающаяся оболочка. От аналогичных продуктов ее отличают гибкая конфигурация, удобные алиасы, модульная структура и самая лучшая система автоматического дополнения. Вы можете сказать, что у bash есть и алиасы, и автодополнение, его можно улучшить с помощью bash-complition, но, дочитав статью до конца, вы убедитесь, что zsh оставляет всех далеко позади. Стоит ли говорить, что zsh заменил мне все файловые менеджеры — как графические (вроде Krusader), так и пресловутый mc? Думаю, начинать лучше постепенно, с первых строчек конфигурации.

```
[~] % cd articles/my/песни\ и\ стихи                    [17:45]
[~/articles/my/песни и стихи] % vim песочная\ кровь      [17:46]
20\ \(жертва\ двадцати\ стен\)
беги\ от\ судьбы
выжигай\ на\ сердце
зачем\ жить\?
зов\ Ктулху
красные\ слёзы\ горгульи
ледяная\ могила
летаргия
лики\ смерти
мясорубка
осколки\ ЗАВТРАка
пассивный\ каннибализм\ \(теория\ самоуничтожения\)
песочная\ кровь
последовательность
пыль
ржавое\ небо
сколько\ стоит\ твоя\ душа\?
спасибо\ тебе\ за\ повод\ умереть
теледильдоника
это\ часть\ меня
я\ иду...
```

Перемещаемся по вариантам дополнений

Приглашение

Главный конфигурационный файл .zshrc должен храниться в вашем домашнем каталоге. Есть также файлы .zlogin и .zlogout, которые выполняются во время входа и выхода из системы соответственно.

Так как многие системные переменные, установленные поставщиком дистрибутива, находятся в файле /etc/profile, его можно запустить с помощью команды source /etc/profile. Как уже замечалось, zsh имеет модульную структуру, и нужные модули также следует загрузить:

```
autoload -U compinit
compinit
autoload -U incremental-complete-word
```

bash и не снились такие PS1

```
zle -N incremental-complete-word
autoload -U insert-files
zle -N insert-files
autoload -U predict-on
zle -N predict-on
```

Теперь можно приступать к настройке PS1. Вообще, в zsh традиционно принято использовать переменную PROMPT, но PS1 тоже будет работать.

Вот основные переменные, которые могут быть использованы в PROMPT:

- %{\e[1;32m%} — цветовой блок; код цвета — как и у bash, в данном случае зеленый;
- %M — полное имя хоста машины;
- %m — имя хоста без домена;
- %n — имя пользователя;
- %~ — текущая директория;
- %T — текущее время в 24-часовом формате.

Если вам этого мало (а так, скорее всего, и есть), посмотрите документацию. Стоит обратить внимание на то, что у zsh может быть так называемый «двусторонний» PROMPT! У меня, к примеру, с левой стороны отображается текущая директория, а с правой — время:

```
PROMPT=$'%{\e[1;32m%}[%{\e[1;34m%}%~%{\e[1;32m%
}] %{\e[1;31m%}%#%{\e[0m%} '
RPROMPT=$'%{\e[1;32m%}[%{\e[1;33m%}%T%{\e[1;32m
%}]%{\e[0m%}'
```

Все это, конечно, замечательно, но большинство пользователей почти всегда работают в графической среде, и было бы не плохо отображать некоторую информацию в заголовке окна терминала. К примеру, помимо пути вы можете захотеть наблюдать текущую команду, выполняемую в терминале. Нет проблем! В zsh присутствуют две специальные функции: выполняющаяся перед каждой командой precmd, а также preexec, которая будет вызвана после выполнения команды. Ими мы как раз и воспользуемся:

```
precmd()
{
        [[ -t 1 ]] || return
        case $TERM in
                *xterm*|rxvt|(dt|k|E)term*) print -Pn
"\e]2;[%~] :: %l\a"
                ;;
        esac
}
preexec() {
        [[ -t 1 ]] || return
        case $TERM in
                *xterm*|rxvt|(dt|k|E)term*) print -Pn
"\e]2;<$1> [%~] :: %l\a"
                ;;
        esac
}
```

История

Нет, в этой рубрике вы не найдете рассказ о том, как создавался zsh. Речь пойдет о настройке истории команд.

Определить файл истории можно следующим образом:

```
HISTFILE=~/.zhistory
```

Число команд, сохраняемых в HISTFILE:

```
SAVEHIST=5000
```

Число команд сохраняемых в сеансе:

```
HISTSIZE=5000
```

Лично мне этих значений хватает, а вот историю нужно еще включить:

```
setopt APPEND_HISTORY
```

Меня не удовлетворяют повторяющиеся команды, лишние пробелы и пустые строки в истории. Эта проблема устраняется следующим образом:

```
setopt HIST_IGNORE_ALL_DUPS
setopt HIST_IGNORE_SPACE
setopt HIST_REDUCE_BLANKS
```

Алиасы

Обычными алиасами никого не удивишь, тем не менее приведу пару примеров:

```
alias mkisofs='mkisofs -J -joliet-long -l -max-iso9660-file-
names'
alias cdrecord='cdrecord -v -dev=1,0,0'
```

Теперь самое интересное. В любом файловом менеджере можно привязать программу к расширению или MIME-типу. Из-за этого многие пользуются слабеньким mc или же очень «тяжелыми» (в плане используемых ресурсов) Konqueror и Nautilus и не хотят попробовать что-то другое. Но zsh тоже умеет назначать алиасы по расширению:

```
alias -s {avi,mpeg,mpg,mov,m2v}=mplayer
alias -s {odt,doc,sxw,rtf}=openoffice.org
alias -s {ogg,mp3,wav,wma}=beep-media-player
alias -s pdf=xpdf
```

Все, теперь bash точно начинает сдавать позиции. А вот для браузеров разработчики написали отдельный модуль:

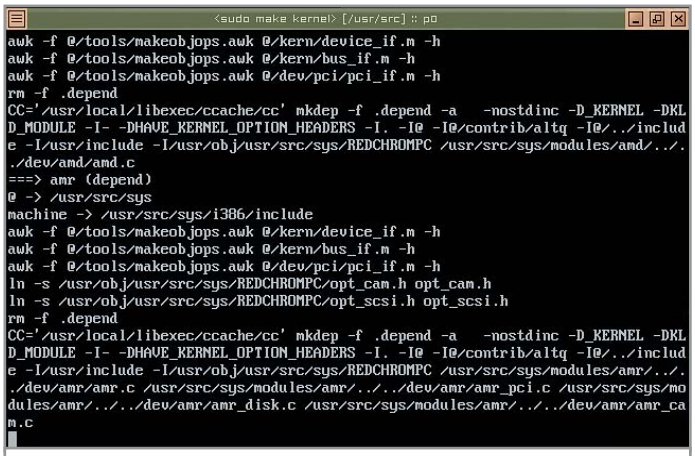
```
autoload -U pick-web-browser
alias -s {html,htm}=pick-web-browser
```

Скрипт поддерживают такие браузеры как Opera, Konqueror, Lynx, Mozilla и другие. Думаю, вы сами при желании сможете найти этот скрипт и заменить Mozilla на любимый Firefox.

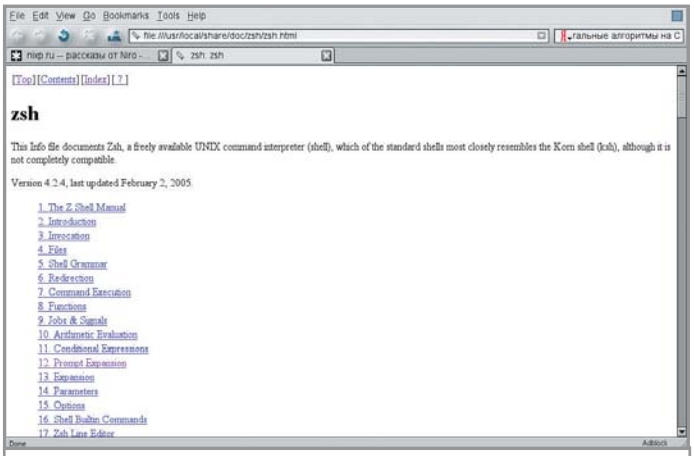
Дополнение

Теперь перейдем к самому интересному. Ни один shell не имеет такой сильной поддержки автодополнений, как zsh. Стандартные функции, заключающиеся в дополнении имен директорий и команд, есть у всех, а как насчет опций определенной программы?

Bash имеет такую возможность, но только за счет установки пакета bash-complition, который будет достаточно сильно



Сигнал с микрофона слишком слабый, поэтому придется немного с ним поработать



Как ни крути, чтобы воспользоваться всей мощью zsh, придется читать документацию

тормозить саму оболочку. Zsh уже «из коробки» может дополнять команды и директории, опции огромного количества программ и сразу представлять все это в виде меню! Я уже не говорю про гибкость настройки системы дополнений и скорость, с которой работает сама оболочка, несмотря на потрясающую функциональность. Как всегда не забываем подгрузить некоторые необходимые модули:

```
zmodload -a zsh/stat stat
zmodload -a zsh/zpty zpty
zmodload -a zsh/zprof zprof
zmodload -ap zsh/mapfile mapfile
```

А теперь можно смело переходить к непосредственной настройке системы дополнений. Чтобы определить опции нужной функциональности, необходимо вызвать функцию zstyle с указанием подсистемы и ее параметрами.

Скучно смотреть на монохромную консоль, в то время как команда ls, например, может подсвечивать все объекты файловой системы разными цветами. Этого можно добиться и с помощью zsh — благодаря опции list-colors, которая принимает тот же синтаксис указания цветов, что и GNU ls. Следовательно, можно использовать переменную $LS_COLORS:

```
zstyle ':completion:*:default' list-colors '${LS_COLORS}'
```

Но так как я работаю во FreeBSD, а установленная там утилита ls использует другой синтаксис, пришлось непосредственно указать цвета:

```
zstyle ':completion:*:default' list-colors
'no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:do=01;3
5:bd=40;33;01:cd=40;33;01:or=40;31;01:ex=01;31:'
```

Далее выберем некоторые опции, которые будут использоваться по умолчанию:

```
zstyle ':completion:*' completer _complete _list _oldlist _
expand _ignored _match _correct _approximate _prefix
zstyle ':completion:*' insert-unambiguous true
zstyle ':completion:*' add-space true
```

Дополнения есть и к командам kill и killall. Они идут под общей подсистемой processes. К примеру, можно указать команду для получения списка процессов:

```
zstyle ':completion:*:processes' command 'ps -xuf'
zstyle ':completion:*:processes-names' command 'ps xho command'
```

Или же отключить их сортировку:

```
zstyle ':completion:*:processes' sort false
```

Если хочется перейти в другую директорию, находящуюся, к примеру, в предыдущей, то наличие в вариантах текущей директории нежелательно:

```
zstyle ':completion:*:cd:*' ignore-parents parent pwd
```

А вот таким образом можно ограничить число ошибок на каждый символ дерева дополнений до одной:

```
zstyle -e ':completion:*:approximate:*' max-errors 'reply=( $((
($#PREFIX+$#SUFFIX)/3 )) numeric )'
```

Для повышения скорости работы пользователя лучше отключить различие регистров:

```
zstyle ':completion:*' matcher-list 'm:{a-z}={A-Z}'
```

Можно и игнорировать внутренние функции дополнений:

```
zstyle ':completion:*:functions' ignored-patterns '_*'
```

Как я уже говорил, можно включить меню выбора вариантов дополнения. К примеру, вы хотите открыть один из нескольких файлов, а они на русском языке — лень переключать раскладку. Просто пишете команду (к примеру, vim), затем пробел, два раза нажимаете кнопку «Tab» — и можете перемещаться по вариантам (в данном случае — файлам) клавишами стрелок или с помощью той же «Tab» и выбирать их! Вот несколько строк, которые позволяют все это проделать:

```
zstyle ':completion:*' menu select=long-list select=0
zstyle ':completion:*' old-menu false
zstyle ':completion:*' original true
zstyle ':completion:*' substitute 1
zstyle ':completion:*' use-compctl true
zstyle ':completion:*' verbose true
zstyle ':completion:*' word true
```

Напоследок

Возможно, у вас возникнут проблемы с распознаванием клавиш. Это решается простым переназначением клавиш в zsh. Почитайте по этому поводу документацию.

Надеюсь, теперь у вас не осталось сомнений в том, что лучше оболочки, чем zsh, попросту не существует, что не обязательно использовать файловые менеджеры для удобства работы, а стремление к аскетизму может привести к удобству.