ZSH-LOVERS(1)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 NAME

zsh-lovers - tips, tricks and examples for the Z shell

2 SYNOPSIS

Just read it. ;-)

3 OVERVIEW

Whenever we look at the zsh manual we wonder why there are no examples or those simply things in (shell) life. The zsh contains many features, but there was no manpage with some examples (like procmailex(5)). That's why we wrote this manpage.

Most of the tricks and oneliner come from the mailinglists zsh-users, zsh-workers, google, newsgroups and from ourself. See section **LINKS** for details.

Note: This manpage (zsh-lovers(1)) is **not** an official part of the Z shell! It's just a just for fun - manpage ;)
For comments, bugreports and feedback take a quick look at the section **BUGS**.

4 SHELL-SCRIPTING

This section provides some examples for often needed shellscript-stuff. Notice that you should not use otherwise most examples won't work.
Parse options in shellscripts. Example taken from ZWS by Adam Chodorowski (http://www.chodorowski.com/projects/zws/):

```
parse_options()
{
    o_port=(-p 9999)
    o_root=(-r WWW)
    o_log=(-d ZWS.log)

    zparseopts -K -- p:=o_port r:=o_root l:=o_log h=o_help
    if [[ $? != 0 || "$o_help" != "" ]]; then
        echo Usage: $(basename "$0") "[-p PORT] [-r DIRECTORY]"
        exit 1
    fi

    port=$o_port[2]
    root=$o_root[2]
    log=$o_log[2]

    if [[ $root[1] != '/' ]]; then root="$PWD/$root"; fi
}
# now use the function:
parse_options $*
```

5 EXAMPLES

Available subsections are **Aliases**, **Completion**, **Unsorted/Misc examples**, **(Recursive) Globbing - Examples**, **Modifiers usage**, **Redirection-Examples**, **ZMV-Examples** and **Module-Examples**.

5.1 ALIASES

Suffix aliases are supported in zsh since version 4.2.0. Some examples:

```
alias -s tex=vim
alias -s html=w3m
alias -s org=w3m
```

Now pressing return-key after entering *foobar.tex* starts vim with foobar.tex. Calling a html-file runs browser w3m. *www.zsh.org* and pressing enter starts w3m with argument www.zsh.org.
Global aliases can be used anywhere in the command line. Example:

```
$ alias -g C='| wc -l'
$ grep alias ~/.zsh/* C
443
```

Some more or less useful global aliases (choose whether they are useful or not for you on your own):

```
alias -g ...='../..'
alias -g ....='../../..'
alias -g .....='../../../..'
alias -g CA="2>&1 | cat -A"
alias -g C='| wc -l'
alias -g D="DISPLAY=:0.0"
alias -g DN=/dev/null
alias -g ED="export DISPLAY=:0.0"
alias -g EG='|& egrep'
alias -g EH='|& head'
alias -g EL='|& less'
alias -g ELS='|& less -S'
alias -g ETL='|& tail -20'
alias -g ET='|& tail'
alias -g F=' | fmt -'
alias -g G='| egrep'
alias -g H='| head'
alias -g HL='|& head -20'
alias -g Sk="*~(*.bz2|*.gz|*.tgz|*.zip|*.z)"
alias -g LL="2>&1 | less"
alias -g L="| less"
alias -g LS='| less -S'
alias -g MM='| most'
alias -g M='| more'
alias -g NE="2> /dev/null"
alias -g NS='| sort -n'
alias -g NUL="> /dev/null 2>&1"
alias -g PIPE='|'
alias -g R=' > /c/aaa/tee.txt '
alias -g RNS='| sort -nr'
alias -g S='| sort'
alias -g TL='| tail -20'
alias -g T='| tail'
alias -g US='| sort -u'
alias -g VM=/var/log/messages
alias -g X0G='| xargs -0 egrep'
alias -g X0='| xargs -0'
alias -g XG='| xargs egrep'
alias -g X='| xargs'
```

5.2 COMPLETION

See also man 1 zshcompctl zshcompsys zshcompwid. zshcompctl is the old style of zsh programmable completion, zshcompsys is the new completion system, zshcompwid are the zsh completion widgets.

Some functions, like _apt and _dpkg, are very slow. You can use a cache in order to proxy the list of results (like the list of available debian packages) Use a cache:

```
zstyle ':completion:*' use-cache on
zstyle ':completion:*' cache-path ~/.zsh/cache
```

Prevent CVS files/directories from being completed:

```
zstyle ':completion:*:(all-|)files' ignored-patterns '(|*/)CVS'
zstyle ':completion:*:cd:*' ignored-patterns '(*/)#CVS'
```

Fuzzy matching of completions for when you mistype them:

```
zstyle ':completion:*' completer _complete _match _approximate
zstyle ':completion:*:match:*' original only
zstyle ':completion:*:approximate:*' max-errors 1 numeric
```

And if you want the number of errors allowed by _approximate to increase with the length of what you have typed so far:

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

Ignore completion functions for commands you don't have:

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

With helper functions like:

```
xdvi() { command xdvi ${*:-*.dvi(om[1])} }
```

you can avoid having to complete at all in many cases, but if you do, you might want to fall into menu selection immediately and to have the words sorted by time:

```
zstyle ':completion:*:*:xdvi:*' menu yes select
zstyle ':completion:*:*:xdvi:*' file-sort time
```

Completing process IDs with menu selection:

```
zstyle ':completion:*:*:kill:*' menu yes select
zstyle ':completion:*:kill:*'   force-list always
```

If you end up using a directory as argument, this will remove the trailing slash (useful in ln)

```
zstyle ':completion:*' squeeze-slashes true
```

cd will never select the parent directory (e.g.: cd ../<TAB>):

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

Another method for *quick change directories*. Add this to your ~/.zshrc, then just enter "cd . . . ./dir"

```
rationalise-dot() {
  if [[ $LBUFFER = *.. ]]; then
    LBUFFER+=/..
  else
    LBUFFER+=.
  fi
}
zle -N rationalise-dot
bindkey . rationalise-dot
```

5.3 UNSORTED/MISC examples

Hint: A list of valid glob Qualifiers can be found in zshexpn(1). See "man 1 zshexpn | less -p" Qualifiers for details.

```
# Get the names of all files that *don't* match a pattern *anywhere* on the
# file (and without ``-L'' because its GNUish)
  $ print -rl -- *(.^e{'grep -q pattern $REPLY'})
  # or
  $ : *(.e{'grep -q pattern $REPLY || print -r -- $REPLY'})

# random numbers
  $ echo $[${RANDOM}%1000]     # random between 0-999
  $ echo $[${RANDOM}%11+10]    # random between 10-20
  $ echo ${(l:3::0:)${RANDOM}} # N digits long (3 digits)

# reverse a word
  $ echo "${(j::)${(@Oa)${(s::):-hello}}}"

# Show newest directory
  $ ls -ld *(/om[1])

# random array element
  $ FILES=( .../files/* )
  $ feh $FILES[$RANDOM%$#FILES+1]

# cat first line in all files in this dir
  $ for file (*(ND-.)) IFS= read -re < $file

# test if a parameter is numeric
  $ if [[ $1 == <-> ]] ; then
        echo numeric
    else
        echo non-numeric
    fi

# Show me all the .c files for which there doesn't exist a .o file.
  $ print *.c(e_'[[ ! -e $REPLY:r.o ]]'_)

# All files in /var/ that are not owned by root
  $ ls -ld /var/*(^u:root)

# All files for which the owner hat read and execute permissions
  $ echo *(f:u+rx:)

# The same, but also others don't have execute permissions
  $ echo *(f:u+rx,o-x:)

# brace expansion - example
  $ X=(A B C)
  $ Y=(+ -)
  $ print -r -- $^X.$^Y
  A.+ A.- B.+ B.- C.+ C.-

# Fetch the newest file containing the string 'fgractg*.log' in the
# filename and contains the string 'ORA-' in it
  $ file=(fgractg*.log(Nm0om[1]))
  $ (($#file)) && grep -l ORA- $file
  # without Zsh
  $ files=$( find . -name . -o -prune -name 'fgractg*>log' -mtime 0 -print )
  > if [ -n "$files" ]; then
  >     IFS='
  > '
```

```
  > set -f
  > file=$(ls -td $files | head -1)
  > grep -l ORA- "$file"
  > fi

# keep specified number of child processes running until entire task finished
  $ zsh -c 'sleep 1 & sleep 3 & sleep 2& print -rl -- $jobtexts'

# Remove zero length and .bak files in a directory
  $ rm -i *(.L0) *.bak(.)

# print out files that don't have extensions
  $ printf '%s\n' ^?*.*
  $ printf '%s\n' ^?*.[^.]*(D)
  $ ls -d -- ^?*.*(D)

# Finding files which does not contain a specific string
  $ print -rl file* | comm -2 -3 - <(grep -l string file*)'
  $ for f (file*(N)) grep -q string $f || print -r $f'

# Show/Check whether a option is set or not. It works both with $options as
# with $builtins
  $ echo $options[correct]
  off
  $ $options[zle]
  on

# Count the number of directories on the stack
  $ print $((${${(z)${(f)"$(dirs -v)"}[-1]}[1]} + 1)) # or
  $ dirs -v | awk '{n=$1}END{print n+1}'

# Matching all files which do not have a dot in filename
  $ ls *~*.*(.)

# Show only the ip-address from ``ifconfig device''
  # ifconfig from net-tools (Linux)
  $ print ${${${$(LC_ALL=C /sbin/ifconfig eth0)[7]}:gs/addr://}
  # ifconfig from 4.2BSD {Free,Net,Open}BSD
  $ print ${$(/sbin/ifconfig tun0)[6]}

# Ping all the IP addresses in a couple of class C's or all hosts
# into /etc/hosts
  $ for i in {1..254}; do ping -c 1 192.168.13.$i; done
  or
  $ I=1
  $ while ( [[ $I -le 255 ]] ) ; do ping -1 2 150.150.150.$I; let I++; done
  or
  $ for i in $(sed 's/#.*//' > /etc/hosts | awk '{print $2}')
  : do
  :    echo "Trying $i ... "
  :    ping -c 1 $i ;
  :    echo '============================='
  : done

# load all available modules at startup
  $ typeset -U m
  $ m=()
  $ for md ($module_path) m=($m $md/**/*(*e:'REPLY=${REPLY#$md/}'::r))
  $ zmodload -i $m

# Rename all files within a directory such that their names get a numeral
# prefix in the default sort order.
```

```
  $ i=1; for j in *; do mv $j $i.$j; ((i++)); done
  $ i=1; for f in *; do mv $f $(echo $i | \
    awk '{ printf("%03d", $0)}').$f; ((i++)); done
  $ integer i=0; for f in *; do mv $f $[i+=1].$f; done

# Find (and print) all symbolic links without a target within the current
# dirtree.
  $ $ file **/*(D@) | fgrep broken
  $ for i in **/*(D@); [[ -f $i || -d $i ]] || echo $i
  $ echo **/*(@-^./=%p)
  $ print -l **/*(-@)

# List all plain files that do not have extensions listed in `fignore'
  $ ls **/*~*(${~${(j/|/)fignore}})(.)
  # see above, but now omit executables
  $ ls **/*~*(${~${(j/|/)fignore}})(.^*)

# Print out files that don't have extensions (require *setopt extendedglob*
# and *setopt dotglob*)
  $ printf '%s\n' ^?*.*

# List files in reverse order sorted by name
  $ print -rl -- *(On)
  or
  $ print -rl -- *(^on)

# Synonymic to ``ps ax | awk '{print $1}'''
  $ print -l /proc/*/cwd(:h:t:s/self//)

# Get the PID of a process (without ``ps'', ``sed'', ``pgrep'', ..
# (under Linux)
  $ pid2 () {
  >   local i
  >   for i in /proc/<->/stat
  > do
  >   [[ "$(< $i)" = *\((${(j:|:)~@})\)* ]] && echo $i:h:t
  > done
  > }

# for X in 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y'; do ...
  $ for (( i = 36#n; i <= 36#y; i++ )); do
  >   print ${$(([##36]i)):l}
  > done
# or in combination with ``dc''
  $ print {$((##n))..$((##y))}P\ 10P | dc
# or with ``eval''
  $ eval print '${$(([##36]'{$((36#n))..$((36#y))}')):l}'

# foreach in one line of shell
  $ for f (*) print -r -- $f

# copy a directory recursively without data/files
  $ dirs=(**/*(/))
  $ cd -- $dest_root
  $ mkdir -p -- $dirs
# or without zsh
  $ find . -type d -exec env d="$dest_root" \
    sh -c ' exec mkdir -p -- "$d/$1"' '{}' '{}' \;

# If `foo=23'', then print with 10 digit with leading '0'.
  $ foo=23
  $ print ${(r:10::0:)foo}
```

```
# find the name of all the files in their home directory that have
# 20 or more characters in their file names
  print -rl $HOME/${(l:20::?:)~:-}*

# Save arrays
  $ print -r -- ${(qq)m} > $nameoffile       # save it
  $ eval "m=($(cat -- $nameoffile)"           # or use
  $ m=("${(@Q)${(z)"$(cat -- $nameoffile)"}}") # to restore it

# get a "ls -l" on all the files in the tree that are younger than a
# specified age (e.g "ls -l" all the files in the tree that where
# modified in the last 2 days)
  $ ls -tld **/*(m-2)
# This will give you a listing 1 file perl line (not à la ls -R).
# Think of an easy way to have a "ls -R" style output with
# only files newer than 2 day old.
  $ for d (. ./**/*(/)) {
  >   print -r -- $'\n'${d}:
  >   cd $d && {
  >       l=(*(Nm-2))
  >       (($#l)) && ls -ltd -- $l
  >       cd ~-
  >   }
  > }
# If you also want directories to be included even if their mtime
# is more than 2 days old:
  $ for d (. ./**/*(/)) {
  >   print -r -- $'\n'${d}:
  >   cd $d && {
  >       l=(*(N/,m-2))
  >       (($#l)) && ls -ltd -- $l
  >       cd ~-
  >   }
  > }
# And if you want only the directories with mtime < 2 days to be listed:
  $ for d (. ./**/*(N/m-2)) {
  >   print -r -- $'\n'${d}:
  >   cd $d && {
  >       l=(*(Nm-2))
  >       (($#l)) && ls -ltd -- $l
  >       cd ~-
  >   }
  > }

# print 42 ``-''
  $ echo ${(l:42::-:)}
# or use ``$COLUMS''
  $ echo ${(l:$COLUMNS::-:)}
# and now with colors (require autoload colors ;colors)
  $ echo "$bg[red]$fg[black]${(l:42::-:)}"

# Redirect STDERR to a command like xless without redirecting STDOUT as well.
  $ foo 2>>(xless)
# but this executes the command asynchronously. To do it synchronously:
  $ { { foo 1>&3 } 2>&1 | xless } 3>&1

# Rename all MP3-Files from name with spaces.mp3 to Name With Spaces.mp3
  $ for i in *.mp3; do
  >     mv $i ${${(C)i}:s/Mp3/mp3/}
  > done
```

```
# Match file names containing only digits and ending with .xml (require
# *setopt kshglob*)
  $ ls -l [0-9]##.xml
  $ ls -l <0->.xml

# Remove all "non txt" files
  $ rm ./^*.txt

# Move 200 files from a directory into another
  $ mv -- *([1,200]) /another/Dir

# Convert images (foo.gif => foo.png):
  $ for i in **/*.gif; convert $i $i:r.png

# convert a collection of mp3 files to wave or cdr,
# e.g. file.wav -> file.mp3)
  $ for i (./*.mp3){mpg321 --w - $i > ${i:r}.wav}

# Download with LaTeX2HTML  created Files (for example the ZSH-Guide):
  $ for f in https://zsh.sourceforge.io/Guide/zshguide{,{01..08}}.html; do
  >     lynx -source $f >${f:t}
  > done

# Move all files in dir1 and dir2 that have line counts greater than 10 to
# another directory say "/more10"
  $ mv dir[12]/**/*.cr(-.e{'((`wc -l < $REPLY` > 10))'}) /more10

# Make with dpkg a master-list of everyfile that it has installed
  $ diff <(find / | sort) <(cat /var/lib/dpkg/info/*.list | sort)

# Replace this fucking Escape-Sequences:
  $ autoload colors ; colors
  $ print "$bg[cyan]$fg[blue]You are a idiot" >> /dev/pts/3

# Get ASCII value of a character
  $ char=N ; print $((#char))

# Filename "Erweiterung"
# Note: The (N) says to use the nullglob option for this particular
# glob pattern.
  $ for i in *.o(N); do
  >     rm $i
  > done

# Rename files; i. e. FOO to foo and bar to BAR
  $ for i in *(.); mv $i ${i:l} # `FOO' to `foo'
  $ for i in *(.); mv $i ${i:u} # `bar to `BAR'

# Show all suid-files in $PATH
  $ ls -latg ${(s.:.)PATH} | grep '^...s'
# or more complex ;)
  $ print -l ${^path}/*(Ns,S)
# or show only executables with a user given pattern
  $ print -l ${^path}/*vim*(*N)

# gzip files when containing a certain string
  $ gzip ${(ps:\0:)"$(grep -lZ foobar ./*.txt(.))"}

# A small  one-liner, that reads from stdin and prints to stdout the first
# unique line i. e. does not print lines that have been printed before
# (this is similar to the unique command, but unique can only handle
# adjacent lines).
```

```
  $ IFS=$'\n\n'; print -rl -- ${(Oau)${(Oa)$(cat file;echo .)[1,-2]}}

# Lists every executable in PATH
  $ print -l ${^path}/*(-*N)

# Match all .c files in all subdirectories, _except_ any SCCS subdirectories?
  $ ls **/*.c~(*/)#SCCS/*

# List all `README' - files case-insensitive with max. one typo
  $ ls **/*(#ia2)readme

# case insensitive checking for variables
  $ if [[ $OSTYPE == (#i)LINUX*(#I) ]]; then
  >    echo "Penguin on board."
  > else
  >    echo "Not a Linux."
  > fi
```

5.4   (Recursive) Globbing - Examples

A list of valid glob Qualifiers can be found in zshexpn(1). **Note:** **/ is equivalent to (*/)#! For example:

```
$ print (*/)#zsh_us.ps
zsh-4.2.3/Doc/zsh_us.ps
$ print **/zsh_us.ps
zsh-4.2.3/Doc/zsh_us.ps
```

```
# Search for `README' in all Subdirectories
  $ ls -l **/README

# find directories that contain both "index.php" and "index.html", or in
# general, directories that contain more than one file matching "index.*"
  $ ls **/*(D/e:'[[ -e $REPLY/index.php && -e $REPLY/index.html ]]':)
  # or
  $ ls **/*(D/e:'l=($REPLY/index.*(N)); (( $#l >= 2 ))':)

# Find command to search for directory name instead of basename
  $ print -rl /**/*~^*/path(|/*)
  # or - without Zsh
  $ find / | grep -e /path/ -e '/path$'

# Print he path of the directories holding the ten biggest C regular files
# in the current directory and subdirectories.
  $ print -rl -- **/*.c(D.OL[1,10]:h) | sort -u

# Find files with size == 0 and send a mail
  $ files=(**/*(ND.L0m+0m-2))
  > (( $#files > 0 )) && print -rl -- $files | \
    mailx -s "empty files" foo@bar.tdl

# recursive chmod
  $ chmod 700 **/(.) # Only files
  $ chmod 700 **/(/) # Only directories

# print out all of the files in that directory in 2 columns
  $ print -rC2 -- ${1:[...]}/*(D:t)
#             ^- number ob columns
# or - if you feel concerned about special characters - use
  $ list=(${1:[...]}/*(ND:t))
  $ (($#list)) && print -rC2 -- ${(V)list}
```

```
# Search all files in /home/*/*-mail/ with a setting ``chmod -s'' flag
# (recursive, include  dotfiles) remove the setgid/setuid flag and print
# a message
  $ chmod -s /home/*/*-mail(DNs,S) /home/*/*-mail/**/*(DNs,S))
# or with a small script
  $ for file (/home/*/*-mail(DNs,S) /home/*/*-mail/**/*(DNs,S)) {
  >    print -r -- $file
  >    chmod -s $file && print -r fixed $file
  > }
# or use ``zargs'' (require autoload zargs) prevent the arg list too
# long error
  $ zargs /home/*/*-mail(DNs,S) /home/*/*-mail/**/*(DNs,S)) -- chmod -s

# List files beginning at `foo23' upwards (foo23, foo24, foo25, ..)
  $ ls -l foo<23->

# get all files that begin with the date strings from June 4 through
# June 9 of 2004
  $ ls -l 200406{04..10}*(N)
# or if they are of the form 200406XX (require ``setopt extended_glob''
  $ ls -l 200306<4-10>.*

# remove spaces from filenames
  $ for a in ./**/*\ *(Dod); do mv $a ${a:h}/${a:t:gs/ /_}; done

# Show only all *.c and *.h - Files
  $ ls -l *.(c|h)

# Show only all *.c - files and ignore `foo.c'
  $ ls *.c~foo.c

# show data to *really* binary format
  $ zsh -ec 'while {} {printf %.8x $n;repeat 8 \
  > {read -ku0 a printf \ %.8d $(([##2]#a))};print;((n+=8))}' < binary

# Show only world-readable files
  $ ls -l *(R)

# List files in the current directory are not writable by the owner
  $ print -l ~/*(ND.^w)

# find and delete the files which are older than a given parameter
# (seconds/minutes/hours)
  # deletes all regular file in /Dir that are older than 3 hours
   $ rm -f /Dir/**/*(.mh+3)
  # deletes all symlinks in /Dir that are older than 3 minutes
   $ rm -f /Dir/**/*(@mm+3)
  # deletes all non dirs in /Dir that are older than 30 seconds
   $ rm -f /Dir/**/*(ms+30^/)
  # deletes all folders, sub-folders and files older than one hour
   $ rm ./**/*(.Dmh+1,.DL0)
  # deletes all files more than 6 hours old
   $ rm -f **/*(mh+6)
  # removes all files but the ten newer ones (delete all but last 10
  # files in a directory)
   $ rm ./*(Om[1,-11])
 Note: If you get a arg list too long, you use the builtin rm. For
      example:
  $ zmodload zsh/files ; rm -f **/*(mh+6)
  or use the zargs function:
  $ autoload zargs ; zargs **/*(mh+6) -- rm -f
```

```
# A User's Guide to the Z-Shell /5.9: Filename Generation and Pattern
# Matching find all files in all subdirectories, searching recursively,
# which have a given name, case insensitive, are at least 50 KB large,
# no more than a week old and owned by the root user, and allowing up
# to a single error in the spelling of the name. In fact, the required
# expression looks like this:
  $ ls **/(#ia1)name(LK+50mw-1u0)

# Change the UID from 102 to 666
  $ chown 666 **/*(u102)

# List all files which have not been updated since last 10 hours
  $ print -rl -- *(Dmh+10^/)

# delete only the oldest file in a directory
  $ rm ./*filename*(Om[1])

# Sort the output from `ls -l' by file size
  $ ls -fld *(OL)

# find most recent file in a directory
  $ setopt dotglob ; print directory/**/*(om[1])

# Show only empty files which nor `group' or `world writable'
  $ ls *(L0f.go-w.)

# Find - and list - the ten newest files in directories and subdirs.
# (recursive)
  $ print -rl -- **/*(Dom[1,10])

# Print only 5 lines by "ls" command (like ``ls -laS | head -n 5'').
  $ ls -fl *(DOL[1,5])

# Display the 5-10 last modified files.
  $ print -rl -- /path/to/dir/**/*(D.om[5,10])

# Find all files without a valid owner.
  $ chmod someuser /**/*(D^u:${(j.:u:.)${(f)"$(</etc/passwd)"}%%:*}:)

# Find all the empty directories in a tree.
  $ for f in **/*(/l2); do foo=($f/*(N)); [[ -z $foo ]] && print $f; done
# Note:Since Zsh 4.2.1 the glob qualifier F indicates a non-empty directory.
# Hence *(F) indicates all subdirectories with entries, *(/^F) means all
# subdirectories with no entries.
  $ ls -ld *(/^F)

# Remove empty directories afterwards.
  $ rmdir ./**/*(/od) 2> /dev/null

# Show only files which are owned by group `users'.
  $ ls -l *(G[users])
```

5.5 Modifiers usage

Modifiers are a powerful mechanism that let you modify the results returned by parameter, filename and history expansion. See zshexpn(1) for details.

```
# NOTE: Zsh 4.3.4 needed!
  $ autoload -U age
```

```
# files modified today
  $ print *(e:age today now:)
# files modified since 5 pm
  $ print *(e-age 17:00 now-)
# ... since 5 o'clock yesterda
  $ print *(e-age yesterday,17:00 now-)
# ... from last Christmas before today
  $ print *(e-age 2006/12/25 today-)
# ... before yesterday
  $ print *(e-age 1970/01/01 yesterday-)
# all files modified between the start of those dates
  $ print *(e:age 2006/10/04 2006/10/09:)
# all files modified on that date
  $ print *(e:age 2006/10/04:)
# Supply times.
  $ print *(e-age 2006/10/04:10:15 2006/10/04:10:45-)

# Remove a trailing pathname component, leaving the head. This works like
# `dirname'.
  $ echo =ls(:h)
  /bin

# Remove all leading pathname components, leaving the tail. This works
# like `basename'.
  $ echo =ls(:t)
  ls

# Remove the suffix from each file (*.sh in this example)
   $f:e is $f file extension
   :h --> head (dirname)
   :t --> tail (basename)
   :r --> rest (extension removed)
  $ for f (*.sh) mv $f $f:r

# Remove a filename extension of the form `.xxx', leaving the root name.
  $ echo $PWD
  /usr/src/linux
  $ echo $PWD:t
  linux

# Remove all but the extension.
  $ foo=23.42
  $ echo $foo
  23.42
  $ echo $foo:e
  42

# Print the new command but do not execute it. Only works with history
# expansion.
  $ echo =ls(:h)
  /bin
  $ !echo:p
  $ echo =ls(:h)

# Quote the substituted words, escaping further substitutions.
  $ bar="23'42"
  $ echo $bar
  23'42
  $ echo $bar:q
  23\'42

# Convert the words to all lowercase.
```

```
  $ bar=FOOBAR
  $ echo $bar
  FOOBAR
  $ echo $bar:l
  foobar

# Convert the words to all uppercase.
  $ bar=foobar
  $ echo $bar
  foobar
  $ echo $bar:u
  FOOBAR

# convert 1st char of a word to uppercase
  $ foo="one two three four"
  $ print -r -- "${(C)foo}"
  One Two Three Four
```

5.6 Redirection-Examples

See zshmisc(1) for more information (or less ${^fpath}/zmv(N))

```
# Append `exit 1' at the end of all *.sh - files
  $ echo "exit 1" >> *.sh

# adding files to foobar.tar.gz
  $ eval set =(gunzip < foobar.tar.gz) '
     tar rf $1 additional.txt &&gzip < $1 > foobar.tar.gz'

# Redirect output to a file AND display on screen
  $ foobar >&1 > file1 > file2 > ..

# pipe single output to multiple inputs
  $ zcat foobar.Z >> (gzip -9 > file1.gz) \
     >> (bzip2 -9 > file1.bz2) \
     >> (acb --best > file1.acb)

# Append /etc/services at the end of file `foo' and `bar'
  $ cat /etc/services >> foo >> bar

# Pipe STDERR
  $ echo An error >&2 2>&1 | sed -e 's/A/I/'

# send standard output of one process to standard input of several processes
# in the pipeline
  $ setopt multios
  $ process1 > >(process1) > >(process2)

# initializing a variable and simultaneously keeping terminal output
  $ setopt multios
  $ { a=$(command >&1 >& 3 3 > &- 2>&1);} 3>&1

# redirect stderr two times
  $ setopt multios ; program 2> file2 > file1 2>&1

# Duplicating stdout and stderr to a logfile
  $ exec 3>&1 > logfile 2>&2 2>&1 >&3 3>&-

# redirect stderr (only) to a file and to orig. stderr:
  $ command 2>&2 2>stderr
```

```
# redirect stderr and stdout to separate files and both to orig. stdout:
  $ command 2>&1 1>&1 2>stderr 1>stdout
# redirect stderr and stdout to separate files and stdout to orig. stdout
# AND stderr to orig. stderr:
  $ command 2>&2 1>&1 2>stderr 1>stdout

# More fun with STDERR ;)
  $ ./my-script.sh 2> >(grep -v moron >error.log)|process-output >output.log
  $  echo "Thats STDOUT" >>(sed 's/stdout/another example/' > foobar)
```

5.7 ZMV-Examples (require autoload zmv)

Note: *-n* means no execution (just print what would happen). At

```
# Remove illegal characters in a fat32 file system. Illegal characters are
#   / : ; * ? " < > |
# NOTE: ``-Q'' and (D) is to include hidden files.
  $ unwanted='[:;*?\"<>|]'
  $ zmv -Q "(**/)(*$~unwanted*)(D)" '$1${2//$~unwanted/}'

# Changing part of a filename (i. e. "file-hell.name" -> "file-heaven.name")
  $ zmv '(*)hell(*)' '${1}heaven${2}'
  # or
  $ zmv '*' '$f:s/hell/heaven/'

# remove round bracket within filenames
# i. e. foo-(bar).avi -> foo-bar.avi
  $ zmv '*' '${f//[()]/}'

# serially all files (foo.foo > 1.foo, fnord.foo > 2.foo, ..)
  $ autoload zmv
  $ ls *
  1.c  asd.foo  bla.foo  fnord.foo  foo.fnord  foo.foo
  $ c=1 zmv '*.foo' '$((c++)).foo'
  $ ls *
  1.c  1.foo  2.foo  3.foo  4.foo  foo.fnord

# Rename "file.with.many.dots.txt" by substituting dots (except for the last
# one!) with a space
  $ touch {1..20}-file.with.many.dots.txt
  $ zmv '(*.*)(.*)' '${1//./ }$2'

# Remove the first 4 chars from a filename
  $ zmv -n '*' '$f[5,-1]' # NOTE: The "5" is NOT a mistake in writing!

# Rename names of all files under the current Dir to lower case, but keep
# dirnames as-is.
  $ zmv -Qv '(**/)(*)(.D)' '$1${(L)2}'

# replace all 4th character, which is "1",  with "2" and so on
  $ autoload -U zmv
  $ zmv '(???)1(???[1-4].txt)' '${1}2${2}'

# Remove the first 15 characters from a string
  $ touch 111111111111111{a-z}
  $ autoload zmv
  $ zmv '*' '$f[16,-1]'

# Replace spaces (any number of them) with a single dash in file names
  $ autload zmv
```

```
  $ zmv -n '(**/)(* *)' '$1${2//( #-## #| ##)/-}'
  # or - with Bash
  $ find . -depth -name '* *' -exec bash -c '
  > shopt -s extglob
  > file=$1
  > dir=${file%/*}
  > name=${file##*/}
  > newname=${name//*([ -]) *([ -])/-}
  > mv -i -- "$file" "$Dir/$newname"' {} {} \;

# Clean up file names and remove special characters
  $ autoload zmv
  $ zmv -n '(**/)(*)' '$1${2//[^A-Za-z0-9._]/_}'

# Add *.py to a bunch of python scripts in a directory (some of them end
# in *.py and give them all a proper extension
  $ autoload zmv
  $ zmv -n '(**/)(con*)(#qe,file $REPLY | grep "python script",)' '$1$2.py'

# lowercase all extensions (i. e. *.JPG) incl. subfolders
  $ autoload zmv
  $ zmv '(**/)(*).(#i)jpg' '$1$2.jpg'
  # Or - without Zsh
  $ find Dir -name '*.[jJ][pP][gG]' -print | while read f
  > do
  >      case $f in
  >        *.jpg) ;
  >        *) mv "$f" "${f%.*}.jpg" ;
  >        esac
  > done

# remove leading zeros from file extension
  $ autoload zmv
  $ ls
  filename.001  filename.003  filename.005  filename.007  filename.009
  filename.002  filename.004  filename.006  filename.008  filename.010
  $ zmv '(filename.)0##(?*)' '$1$2'
  $ ls
  filename.1  filename.10  filename.2  filename.3  filename.4  filename.5 ..

# renumber files.
  $ autoload zmv
  $ ls *
  foo_10.jpg  foo_2.jpg  foo_3.jpg  foo_4.jpg  foo_5.jpg  foo_6.jpg ..
  $ zmv -fQ 'foo_(<0->).jpg(.nOn)' 'foo_$(($1 + 1)).jpg'
  $ ls *
  foo_10.jpg  foo_11.jpg  foo_3.jpg  foo_4.jpg  foo_5.jpg  ...

# adding leading zeros to a filename (1.jpg -> 001.jpg, ..
  $ autoload zmv
  $ zmv '(<1->).jpg' '${(l:3::0:)1}.jpg'

# See above, but now only files with a filename >= 30 chars
  $ autoload zmv
  $ c=1 zmv "${(l:30-4::?:)}*.foo" '$((c++)).foo'

# Replace spaces in filenames with a underline
  $ autoload zmv
  $ zmv '* *' '$f:gs/ /_'

# Change the suffix from *.sh to *.pl
  $ autoload zmv
```

```
  $ zmv -W '*.sh' '*.pl'

# Add a "".txt" extension to all the files within ${HOME}
  # ``-.'' is to only rename regular files or symlinks to regular files,
  # ``D'' is to also rename hidden files (dotfiles))
  $ autoload zmv
  $ zmv -Q '/home/**/*(D-.)' '$f.txt'
  # Or to only rename files that don't have an extension:
  $ zmv -Q '/home/**/^?*.*(D-.)' '$f.txt'

# Recursively change filenames with characters ? [ ] / = + < > ; : " , - *
  $ autoload zmv
  $ chars='[][?=+<>;",*-]'
  $ zmv '(**/)(*)' '$1${2//$~chars/%}'

# Removing single quote from filenames (recursively)
  $ autoload zmv
  $ zmv -Q "(**/)(*'*)(D)" "\$1\${2//'/}"

# When a new file arrives (named file.txt) rename all files in order to
# get (e. g. file119.txt becomes file120.txt, file118.txt becomes
# file119.txt and so on ending with file.txt becoming file1.txt
  $ autoload zmv
  $ zmv -fQ 'file([0-9]##).txt(On)' 'file$(($1 + 1)).txt'

# lowercase/uppercase all files/directories
  $ autoload zmv
  $ zmv '(*)' '${(L)1}' # lowercase
  $ zmv '(*)' '${(U)1}' # uppercase

# Remove the suffix *.c from all C-Files
  $ autoload zmv
  $ zmv '(*).c' '$1'

# Uppercase only the first letter of all *.mp3 - files
  $ autoload zmv
  $ zmv '([a-z])(*).mp3' '${(C)1}$2.mp3'

# Copy the target `README' in same directory as each `Makefile'
  $ autoload zmv
  $ zmv -C '(**/)Makefile' '${1}README'

# Removing single quote from filenames (recursively)
  $ autoload zmv
  $ zmv -Q "(**/)(*'*)(D)" "\$1\${2//'/}"

# Rename pic1.jpg, pic2.jpg, .. to pic0001.jpg, pic0002.jpg, ..
  $ autoload zmv
  $ zmv 'pic(*).jpg' 'pic$((l:4::0:)1}.jpg'
  $ zmv '(**/)pic(*).jpg' '$1/pic${(l:4::0:)2}.jpg' # recursively
```

5.8 Module-Examples

Please read zshmodules(1) first!

5.8.1 zsh/pcre (require zmodload zsh/pcre)

```
# Copy files of a certain period (date indicated in the filenames)
  $ zmodload zsh/pcre
```

```
  $ ls -d -- *(e:'[[ $REPLY -pcre-match pcre-regexp ]]':)
  # or
  $ m() { [[ $1 -pcre-match pcre-regexp ]] }
  $ ls -d -- *(+m)
```

5.8.2 zsh/clone (require zmodload zsh/clone)

```
# Creates a forked instance of the current shell ($! is set to zero) and
# execute ``command'' on /dev/tty8 (for this example).
  $ zmodload zsh/clone
  $ clone /dev/tty8 && (($! == 0)) && exec command
```

5.8.3 zsh/datetime (require zmodload zsh/datetime)

```
  $ zmodload zsh/datetime
  $ alias datereplacement='strftime "%Y-%m-%d" $EPOCHSECONDS'
  $ export DATE=`datereplacement`
  $ echo $DATE

#  strip date from filename
  $ $ zmodload zsh/datetime
  $ setopt extendedglob
  $ touch aaa_bbb_20041212_c.dat eee_fff_20051019_g.dat
  $ strftime -s pattern \
    '???_???_<0-%Y%m%d>_?.dat' $((EPOCHSECONDS - 365 * 24 * 60 * 60 / 2))
  $ print -rl -- $~pattern
  aaa_bbb_20041212_c.dat
  $ print -rl -- $pattern
  ???_???_<0-20050815>_?.dat

# Search files size == 0, to be based on the file name containing a date
# rather than the "last modified" date of the file
  $ zmodload -i zsh/datetime
  $ strftime -s file "abc_de_%m%d%Y.dat" $((EPOCHSECONDS - 24 * 60 * 60 ))
  $ files=(**/$file(N.L0))
  $ (( $#files > 0 )) && print -rl -- $files | \
    mailx -s "empty files"  foo@bar.tdl
```

5.8.4 zsh/stat (require zmodload zsh/stat)

```
# test if a symbolic link links to a certain file
  $ zmodload -i zsh/stat
  $ ! stat -LH s foo.ln || [[ $s[link] != "foo.exe" ]] || ln -sf foo.exe foo.ln

# comparing file dates
  $ zmodload zsh/stat
  $ file1=foo
  $ file2=bar
  $ touch bar & sleep 5 & touch foo
  $ echo $file1 is $(($(stat +mtime $file2) - \
    $(stat +mtime $file1))) seconds older than $file2.
  bar is 5 seconds older than foo

# list the files of a disk smaller than some other file
  $ zmodload zsh/stat
  $ stat -A max +size some-other-file
```

```
  $ print -rl ./**/*(D.L-$max)

# List the top 100 biggest files in a disk
  $ zmodload zsh/stat
  $ ls -fld ./**/*(d`stat +device .`OL[1,100])

# Get only the user name and the file names from (like
# ls -l * | awk '{print $3" " $8}')
  $ zmodload zsh/stat
  $ for file; do
  >   stat -sA user +uid -- "$file" &&
  >     print -r -- "$user" "$file"
  > done

# get the difference between actual bytes of file and allocated bytes of file
  $ zmodload zsh/stat
  $ print $(($(stat +block -- file) * 512 - $(stat +size -- file)))

# Find largest file
# ``D''  : to include dot files (d lowercase is for device)
# ``O''  : reverse Ordered (o lowercase for non-reverse order)
# ``L''  : by file Length (l is for number of links)
# ``[1]'': return only first one
  $ zmodload zsh/stat
  $ stat +size ./*(DOL[1])

# file size in bytes
  $ zmodload zsh/stat
  $ stat -L +size ~/.zshrc
  4707

# Delete files in a directory that hasn't been accessed in the last ten days
# and send ONE mail to the owner of the files informing him/her of the files'
# deletion.
  $ zmodload zsh/stat zsh/files
  $ typeset -A f; f=()
  $ rm -f /path/**/*(.a+10e{'stat -sA u +uidr $REPLY; f[$u]="$f[$u]$REPLY"'})
  $ for user (${(k)f}) {print -rn $f[$user]|mailx -s "..." $user}

# Get a "ls -l" on all the files in the tree that are younger than a
# specified age
  $ zmodload zsh/stat
  $ for d (. ./**/*(N/m-2))
  >   print -r -- $'\n'$d: && cd $d && {
  >     for f (*(Nm-2om))
  >     stat -F '%b %d %H:%M' -LsAs -- $f &&
  >     print -r -- $s[3] ${(l:4:)s[4]} ${(l:8:)s[5]} \
  >     ${(l:8:)s[6]} ${(l:8:)s[8]} $s[10] $f ${s[14]:+-> $s[14]}
  >     cd ~-
  >   }

# get file creation date
  $ zmodload zsh/stat
  $ stat -F '%d %m %Y' +mtime ~/.zshrc
  30 06 2004
  $ stat -F '%D' +mtime ~/.zshrc
  06/30/04
```

5.8.5 zsh/files (require zmodload zsh/files)

```
# search a directory for files containing a certain string then copy those
# files to another directory.
  $ zmodload zsh/files
  $ IFS=$'\0'
  $ cp $(grep -lZr foobar .) otherdirectory
```

5.8.6 zsh/mapfile (require zmodload zsh/mapfile)

```
# grepping for two patterns
  $ zmodload zsh/mapfile
  $ pattern1="foo"
  $ pattern2="bar foo"
  $ print -l ./**/*(DN.e{'z=$mapfile[$REPLY] && [[ $z = *$pattern1* && \
    $z = *$pattern2* ]]'})
# or a solution in combination with zsh/pcre
  $ zmodload -i zsh/mapfile zsh/pcre
  $ pattern1="foo"
  $ pattern2="bar foo"
  $ pcre_compile "(?s)(?=.*?$pattern1).*?$pattern2"
  $ pcre_study
  $ print -l ./**/*(DN.e{'pcre_match $mapfile[$REPLY]'})

# equivalent for ``less /etc/passwd | grep -v root''
  $ zmodload zsh/mapfile
  $ IFS=$'\n\n'
  $ print -rl -- ${${=mapfile[/etc/passwd]}:#*root*}
# or - for case insensitive
  $ setopt extendedglob
  $ print -rl -- ${${=mapfile[/etc/passwd]}:#*(#i)root*}

# If a XML-file contains stuff like ``<TAGA/>'' and ``<TAGB/>'', number
# this empty tags (ones ending in '/>') so if encountered in the same
# order, the preceding tags would become ``<TAGA/>1</TAGA>'' and
# ``<TAGB/>2</TAGB>''
  $ zmodload zsh/mapfile
  $ cnt=0
  $ apfile[data.xml.new]=${(S)mapfile[data.xml]//\
  > (#im)<TAGA>*<\/TAGA>/<TAGA>$((++cnt))<\/TAGA>}

# removing all files in users Maildir/new that contain ``filename="gone.src''
  $ zmodload zsh/{files,mapfile}
  $ rm -f /u1/??/*/Maildir/new/100*(.e{'[[ $mapfile[$REPLY] == \
    *filename=\"gone.scr\"* ]]'})

# Grep out the Title from a postscript file and append that value to the
# end of the filename
  $ autoload -U zmv
  $ zmodload zsh/mapfile
  $ zmv '(*).ps' '$1-${${${${mapfile[$f]##*%%Title: }%% *}//[^a-zA-Z0-9_]/}.ps'
```

5.8.7 zsh/mathfunc (require zmodload zsh/mathfunc)

```
$ zmodload zsh/mathfunc
$ echo $(( sin(1/4.0)**2 + cos(1/4.0)**2 - 1 ))
 -1.1102230246251565e-16
$ echo $(( pi = 4.0 * atan(1.0) ))
 3.1415926535897931
```

```
$ echo $(( f = sin(0.3) ))
  0.29552020666133955
$ print $((1e12 * rand48()))
  847909677310.23413
$ print $(( rand48(seed) ))
  0.01043488334700271
```

5.8.8   zsh/termcap (require zmodload zsh/termcap)

```
 $ zmodload -ab zsh/termcap echotc
 $ GREEN=`echotc AF 2`
 $ YELLOW=`echotc AF 3`
 $ RED=`echotc AF 1`
 $ BRIGHTRED=`echotc md ; echotc AF 1`
 $ print -l ${GREEN}green ${YELLOW}yellow ${RED}red ${BRIGHTRED}brightred
```

5.8.9   zsh/zpty (require zmodload zsh/zpty)

```
  $ zmodload zsh/zpty
  $ zpty PW passwd $1
  $ zpty PW passwd $1
# ``-r'': read the output of the command name.
# ``z'' : Parameter
  $ zpty -r PW z '*password:'
# send the to command name the given strings as input
  $ zpty -w PW $2
  $ zpty -r PW z '*password:'
  $ zpty -w PW $2
# The second form, with the -d option, is used to delete commands
# previously started, by supplying a list of their names. If no names
# are given, all commands are deleted. Deleting a command causes the HUP
# signal to be sent to the corresponding process.
  $ zpty -d PW
```

5.8.10   zsh/net/socket (require zmodload zsh/net/socket)

```
# ``-l'': open a socket listening on filename
# ``-d'': argument will be taken as the target file descriptor for the
#         connection
# ``3'' : file descriptor. See ``A User's Guide to the Z-Shell''
#         (3.7.2: File descriptors)
  $ zmodload zsh/net/socket
  $ zsocket -l -d 3
# ``-a'': accept an incoming connection to the socket
  $ zsocket -a -d 4 3
  $ zsocket -a -d 5 3 # accept a connection
  $ echo foobar >&4
  $ echo barfoo >&5
  $ 4>&- 5>&- 3>&
```

5.8.11   zsh/zftp (require zmodload zsh/zftp)

```
  $ autoload -U zfinit
  $ zfinit
  $ zfparams www.example.invalid myuserid mypassword
  $ zfopen
  $ zfcd tips
  $ zfls -l zshtips.html
  $ zfput zshtips.html
  $ zfls -l zshtips.html

# Automatically transfer files using FTP with error checking
  $ autoload -U zfinit ; zfinit
  $ zftp open host.name.invalid user passwd || exit
  $ zftp get /remote/file > /local/file; r=$?
  $ zftp close && exit r

# compress and ftp on the fly
  $ autoload -U zfinit ; zfinit
  $ zftp open host.name.invalid user password
  $ zftp get $file | bzip2 > ${file}.bz2
  $ zftp close

# Recursice ``get''
  $ autoload -U zfinit ; zfinit
  $ zfanon cr.yp.to
  $ zfcd daemontools
  $ for file in `zfls` ; do
  >     zfget $file
  $ done
  $ zfclose

# Upload all regular files in $HOME/foobar (recursive) that are newer than
# two hours to ftp.foobar.invalid/path/to/upload
  $ autoload -U zfinit ; zfinit
  $ zfopen ftp.foobar.invalid/path/to/upload
  $ cd $HOME/foobar
  $ zfput -r **/*(.mh-2)
  $ zfclose

# long list of files on a ftp
  $ autoload -U zfinit ; zfinit
  $ zfopen some-host
  $ zfcd /some/remote/Dir
  $ cd /some/local/Dir
# If the list.txt is located on the remote host, change to
# zfget ${(f)"$(zftp get /path/to/remote/list.txt)"}
  $ zfget ${(f)"$(cat list.txt)"}
  $ zfclose
```

5.8.12   zsh/zselect (require zmodload zsh/zselect)

```
# It's similar to
 ,----
 | $ sg=$(stty -g)
 | $ stty -icanon min 0 time 50
 | $ read yesno
 | $ stty "$sg"
 | $ case "$yesno" in
 | >  yes) command1;;
 | >  *) command2;;
```

```
 | > esac
 `____
$ zmodload zsh/zselect
$ if zselect -t 500 -r 0 && read yesno && [ yes = "$yesno" ]; then
>     command1
> else
>     command1
> fi
```

6 OPTIONS

6.1 Navigation options

auto_cd (allow one to change to a directory by entering it as a command). **auto_pushd** (automatically append dirs to the push/pop list) pushd_ignore_dups (and don't duplicate them).

6.2 Misc

no_hup (don't send HUP signal to background jobs when exiting ZSH). **print_exit_value** (show a message with the exit code when a command returns with a non-zero exit code)

6.2.1 History options

hist_verify (let the user edit the command line after history expansion (e.g. !ls) instead of immediately running it)
Use the same history file for all sessions :
setopt SHARE_HISTORY

6.2.2 Privacy / Security

no_clobber (or set -C; prevent > redirection from truncating the given file if it already exists)

6.2.3 Spelling correction

correct (automatically correct the spelling of commands). **correct_all** (automatically correct the spelling of each word on the command line) **dvorak** (dvorak layout)

7 UNSORTED/MISC

Mailpath: simple multiple mailpath:

```
mailpath=($HOME/Mail/mbox'?new mail in mbox'
          $HOME/Mail/tux.u-strasbg'?new mail in tux'
          $HOME/Mail/lilo'?new mail in lilo'
          $HOME/Mail/ldap-fr'?new mail in ldap-fr')
```

Mailpath: dynamic mailpath:

```
typeset -a mailpath
for i in ~/Mail/Lists/*(.); do
   mailpath[$#mailpath+1]="${i}?You have new mail in ${i:t}."
done
```

Avoid globbing on special commands:

```
for com in alias expr find mattrib mcopy mdir mdel which;
alias $com="noglob $com"
```

For migrating your bashprompt to zsh use the script bash2zshprompt located in the zsh source distribution under *Misc*.

For migration from (t)csh to zsh use the c2z tool that converts csh aliases and environment and shell variables to zsh. It does this by running csh, and having csh report on aliases and variables. The script then converts these to zsh startup files. It has some issues and usage information that are documented at the top of this script.

Here are functions to set the title and hardstatus of an **XTerm** or of **GNU Screen** to *zsh* and the current directory, respectively, when the prompt is displayed, and to the command name and rest of the command line, respectively, when a command is executed:

```
function title {
     if [[ $TERM == "screen" ]]; then
       # Use these two for GNU Screen:
       print -nR $' 33k'$1$' 33'\
       print -nR $' 33]0;'$2$''
     elif [[ $TERM == "xterm" || $TERM == "rxvt" ]]; then
       # Use this one instead for XTerms:
       print -nR $' 33]0;'$*$''
     fi
}
function precmd { title zsh "$PWD" }
function preexec {
    emulate -L zsh
    local -a cmd; cmd=(${(z)1})
    title $cmd[1]:t "$cmd[2,-1]"
}
```

Put the following line into your ~/.screenrc to see this fancy hardstatus:

```
caption always "%3n %t%? (%u)%?%?: %h%?"
```

Special variables which are assigned:

```
$LINENO $RANDOM $SECONDS $COLUMNS $HISTCHARS $UID
$EUID $GID $EGID $USERNAME $fignore $mailpath $cdpath
```

8 LINKS

Primary site
     http://www.zsh.org/

Zsh Web Pages
     https://zsh.sourceforge.io/

Project-page
     http://sourceforge.net/projects/zsh/

From Bash to Z Shell: Conquering the Command Line - the book
     http://www.bash2zsh.com/

"Zsh - die magische Shell" (german book about Zsh) by Sven Guckes and Julius Plenz
     http://zshbuch.org/

Mailinglistarchive
     http://www.zsh.org/mla/

ZSH-FAQ
> https://zsh.sourceforge.io/FAQ/zshfaq.html

Userguide
> https://zsh.sourceforge.io/Guide/

ZSH-Wiki
> http://zshwiki.org/home/

Mouse-Support ;)
> http://stchaz.free.fr/mouse.zsh

ZSH Prompt introduction
> http://aperiodic.net/phil/prompt/

ft's blog posts around zsh
> http://bewatermyfriend.org/tag/zsh/

ft's zsh configuration
> https://gitlab.com/ft/etc-zsh

Adam's ZSH page
> http://www.adamspiers.org/computing/zsh/

Zzappers Best of ZSH Tips
> http://www.rayninfo.co.uk/tips/zshtips.html

Zsh Webpage by Christian Schneider
> http://www.strcat.de/zsh/

The zsh-lovers webpage
> http://grml.org/zsh/

IRC channel
> #zsh at irc.freenode.org

The Z shell reference-card (included in the zsh-lovers debian-package)
> http://www.bash2zsh.com/zsh_refcard/refcard.pdf

9 AUTHORS

This manpage was written by Michael Prokop, Christian *strcat* Schneider and Matthias Kopfermann. But many ideas have been taken from zsh-geeks e.g. from the zsh-mailinglists (zsh-users and zsh-workers), google, newsgroups and the zsh-Wiki. Thanks for your cool and incredible tips. We learned much from you!

In alphabetic order:

```
Andrew 'zefram' Main   - http://www.fysh.org/~zefram/
Barton E. Schaefer     - http://www.well.com/user/barts/
Matthias Kopfermann    - http://www.guckes.net/zsh/lover.html
Oliver Kiddle          - http://people.freenet.de/opk/
Paul Falstad           - http://www.falstad.com/
Peter Stephenson       - http://homepage.ntlworld.com/p.w.stephenson/
Richard Coleman
Stephane Chazelas      - http://stephane.chazelas.free.fr/
Sven Guckes            - http://www.guckes.net/
Sven Wischnowsky       - http://w9y.de/
```

10 SEE ALSO

Manpages of zsh:

```
zsh         Zsh overview
zshall      Tthe Z shell meta-man page
zshbuiltins Zsh built-in commands
zshcalsys   zsh calendar system
zshcompctl  zsh programmable completion
zshcompsys  Zsh completion system
zshcompwid  Zsh completion widgets
zshcontrib  User contributions to zsh
zshexpn     Zsh expansion and substitution
zshmisc     Anything not fitting into the other sections
zshmodules  Zsh loadable modules
zshoptions  Zsh options
zshparam    Zsh parameters
zshroadmap  Informal introduction to the zsh manual
zshtcpsys   Zsh tcp system
zshzle      Zsh command line editing
zshzftpsys  Zsh built-in FTP client
zshall      Meta-man page containing all of the above
```

Note: especially *man zshcontrib* covers very useful topics!
Book: **From Bash to Z Shell** by Oliver Kiddle, Jerry Peck and Peter Stephenson. **ISBN: 1590593766**. - **bash2zsh.com**
Also take a look at the section **LINKS** in this manpage.

11 BUGS

Probably. This manpage might be never complete. So please report bugs, feedback and suggestions via https://github.com/grml/-zsh-lovers Thank you!

12 COPYRIGHT

Copyright (C) Michael Prokop, Christian Schneider and Matthias Kopfermann.