Z: quicker CD

Download z.sh from https://github.com/rupa/z and place it in your home folder. Edit .bashrc and add **. ~/path to z.sh** to it. After next logon, **cd** around for a bit. You should now have a new file in your home folder **.z**, and you can now move around using the **z** command.

Note: You have to enter the folder the old way first

|z.sh

```
# Copyright (c) 2009 rupa deadwyler. Licensed under the WTFPL license,
Version 2

# maintains a jump-list of the directories you actually use
#
# INSTALL:
#     * put something like this in your .bashrc/.zshrc:
#         . /path/to/z.sh
#     * cd around for a while to build up the db
#     * PROFIT!!
#     * optionally:
#         set $_Z_CMD in .bashrc/.zshrc to change the command (default
z).
#         set $_Z_DATA in .bashrc/.zshrc to change the datafile
(default ~/.z).
#         set $_Z_NO_RESOLVE_SYMLINKS to prevent symlink resolution.
#         set $_Z_NO_PROMPT_COMMAND if you're handling PROMPT_COMMAND
yourself.
#         set $_Z_EXCLUDE_DIRS to an array of directories to exclude.
#         set $_Z_OWNER to your username if you want use z while sudo
with $HOME kept
#
# USE:
#     * z foo     # cd to most frecent dir matching foo
#     * z foo bar # cd to most frecent dir matching foo and bar
#     * z -r foo  # cd to highest ranked dir matching foo
#     * z -t foo  # cd to most recently accessed dir matching foo
#     * z -l foo  # list matches instead of cd
#     * z -e foo  # echo the best match, don't cd
#     * z -c foo  # restrict matches to subdirs of $PWD

[ -d "${_Z_DATA:-$HOME/.z}" ] && {
    echo "ERROR: z.sh's datafile (${_Z_DATA:-$HOME/.z}) is a
directory."
}

_z() {

    local datafile="${_Z_DATA:-$HOME/.z}"
```

```
# if symlink, dereference
[ -h "$datafile" ] && datafile=$(readlink "$datafile")

# bail if we don't own ~/.z and $_Z_OWNER not set
[ -z "$_Z_OWNER" -a -f "$datafile" -a ! -O "$datafile" ] && return

_z_dirs () {
    local line
    while read line; do
        # only count directories
        [ -d "${line%%\|*}" ] && echo "$line"
    done < "$datafile"
    return 0
}

# add entries
if [ "$1" = "--add" ]; then
    shift

    # $HOME isn't worth matching
    [ "$*" = "$HOME" ] && return

    # don't track excluded directory trees
    local exclude
    for exclude in "${_Z_EXCLUDE_DIRS[@]}"; do
        case "$*" in "$exclude*") return;; esac
    done

    # maintain the data file
    local tempfile="$datafile.$RANDOM"
    _z_dirs | awk -v path="$*" -v now="$(date +%s)" -F"|" '
        BEGIN {
            rank[path] = 1
            time[path] = now
        }
        $2 >= 1 {
            # drop ranks below 1
            if( $1 == path ) {
                rank[$1] = $2 + 1
                time[$1] = now
            } else {
                rank[$1] = $2
                time[$1] = $3
            }
            count += $2
        }
        END {
            if( count > 9000 ) {
                # aging
                for( x in rank ) print x "|" 0.99*rank[x] "|"
```

```
time[x]
                } else for( x in rank ) print x "|" rank[x] "|" time[x]
            }
        ' 2>/dev/null >| "$tempfile"
        # do our best to avoid clobbering the datafile in a race
condition.
        if [ $? -ne 0 -a -f "$datafile" ]; then
            env rm -f "$tempfile"
        else
            [ "$_Z_OWNER" ] && chown $_Z_OWNER:"$(id -ng $_Z_OWNER)"
"$tempfile"
            env mv -f "$tempfile" "$datafile" || env rm -f "$tempfile"
        fi

    # tab completion
    elif [ "$1" = "--complete" -a -s "$datafile" ]; then
        _z_dirs | awk -v q="$2" -F"|" '
            BEGIN {
                q = substr(q, 3)
                if( q == tolower(q) ) imatch = 1
                gsub(/ /, ".*", q)
            }
            {
                if( imatch ) {
                    if( tolower($1) ~ q ) print $1
                } else if( $1 ~ q ) print $1
            }
        ' 2>/dev/null

    else
        # list/go
        local echo fnd last list opt typ
        while [ "$1" ]; do case "$1" in
            --) while [ "$1" ]; do shift; fnd="$fnd${fnd:+ }$1";done;;
            -*) opt=${1:1}; while [ "$opt" ]; do case ${opt:0:1} in
                    c) fnd="^$PWD $fnd";;
                    e) echo=1;;
                    h) echo "${_Z_CMD:-z} [-cehlrtx] args" >&2;
return;;
                    l) list=1;;
                    r) typ="rank";;
                    t) typ="recent";;
                    x) sed -i -e "\:^${PWD}|.*:d" "$datafile";;
                esac; opt=${opt:1}; done;;
             *) fnd="$fnd${fnd:+ }$1";;
        esac; last=$1; [ "$#" -gt 0 ] && shift; done
        [ "$fnd" -a "$fnd" != "^$PWD " ] || list=1

        # if we hit enter on a completion just go there
        case "$last" in
            # completions will always start with /
```

```
            /*) [ -z "$list" -a -d "$last" ] && builtin cd "$last" &&
return;;
        esac

        # no file yet
        [ -f "$datafile" ] || return

        local cd
        cd="$( < <( _z_dirs ) awk -v t="$(date +%s)" -v list="$list" -v
typ="$typ" -v q="$fnd" -F"|" '
            function frecent(rank, time) {
                # relate frequency and time
                dx = t - time
                if( dx < 3600 ) return rank * 4
                if( dx < 86400 ) return rank * 2
                if( dx < 604800 ) return rank / 2
                return rank / 4
            }
            function output(matches, best_match, common) {
                # list or return the desired directory
                if( list ) {
                    cmd = "sort -n >&2"
                    for( x in matches ) {
                        if( matches[x] ) {
                            printf "%-10s %s\n", matches[x], x | cmd
                        }
                    }
                    if( common ) {
                        printf "%-10s %s\n", "common:", common >
"/dev/stderr"
                    }
                } else {
                    if( common ) best_match = common
                    print best_match
                }
            }
            function common(matches) {
                # find the common root of a list of matches, if it
exists
                for( x in matches ) {
                    if( matches[x] && (!short || length(x) <
length(short)) ) {
                        short = x
                    }
                }
                if( short == "/" ) return
                for( x in matches ) if( matches[x] && index(x, short)
!= 1 ) {
                    return
                }
                return short
```

```
            }
            BEGIN {
                gsub(" ", ".*", q)
                hi_rank = ihi_rank = -9999999999
            }
            {
                if( typ == "rank" ) {
                    rank = $2
                } else if( typ == "recent" ) {
                    rank = $3 - t
                } else rank = frecent($2, $3)
                if( $1 ~ q ) {
                    matches[$1] = rank
                } else if( tolower($1) ~ tolower(q) ) imatches[$1] =
rank
                if( matches[$1] && matches[$1] > hi_rank ) {
                    best_match = $1
                    hi_rank = matches[$1]
                } else if( imatches[$1] && imatches[$1] > ihi_rank ) {
                    ibest_match = $1
                    ihi_rank = imatches[$1]
                }
            }
            END {
                # prefer case sensitive
                if( best_match ) {
                    output(matches, best_match, common(matches))
                } else if( ibest_match ) {
                    output(imatches, ibest_match, common(imatches))
                }
            }
        ')"

        [ $? -eq 0 ] && [ "$cd" ] && {
          if [ "$echo" ]; then echo "$cd"; else builtin cd "$cd"; fi
        }
    fi
}

alias ${_Z_CMD:-z}='_z 2>&1'

[ "$_Z_NO_RESOLVE_SYMLINKS" ] || _Z_RESOLVE_SYMLINKS="-P"

if type compctl >/dev/null 2>&1; then
    # zsh
    [ "$_Z_NO_PROMPT_COMMAND" ] || {
        # populate directory list, avoid clobbering any other precmds.
        if [ "$_Z_NO_RESOLVE_SYMLINKS" ]; then
            _z_precmd() {
                (_z --add "${PWD:a}" &)
            }
```

```
        else
            _z_precmd() {
                (_z --add "${PWD:A}" &)
            }
        fi
        [[ -n "${precmd_functions[(r)_z_precmd]}" ]] || {
            precmd_functions[$(($#precmd_functions+1))]=_z_precmd
        }
    }
    _z_zsh_tab_completion() {
        # tab completion
        local compl
        read -l compl
        reply=(${(f)"$(_z --complete "$compl")"})
    }
    compctl -U -K _z_zsh_tab_completion _z
elif type complete >/dev/null 2>&1; then
    # bash
    # tab completion
    complete -o filenames -C '_z --complete "$COMP_LINE"' ${_Z_CMD:-z}
    [ "$_Z_NO_PROMPT_COMMAND" ] || {
        # populate directory list. avoid clobbering other
PROMPT_COMMANDs.
        grep "_z --add" <<< "$PROMPT_COMMAND" >/dev/null || {
            PROMPT_COMMAND="$PROMPT_COMMAND"$'\n''(_z --add "$(command
pwd '$_Z_RESOLVE_SYMLINKS' 2>/dev/null)" 2>/dev/null &);'
        }
    }
fi
```