```
################################################################################
# Bestand:  README.txt                                                         #
# Doel:     Tekstbestand met toelichting op de scripts.                        #
# Gebruik:  Toelichting op eigengemaakte scripts.                              #
# Gebruikt: -                                                                  #
# Auteur:   Karel Zimmer (http://karelzimmer.nl, info@karelzimmer.nl)          #
# ---------------------------------------------------------------------------- #
# Dit werk valt onder een Creative Commons Naamsvermelding-GelijkDelen 4.0     #
# Internationaal licentie (CC BY-SA 4.0).                                      #
# Bezoek http://creativecommons.org/licenses/by-sa/4.0/deed.nl om een kopie    #
# te zien van de licentie of stuur een brief naar Creative Commons,            #
# PO Box 1866, Mountain View, CA 94042, USA.                                   #
# ---------------------------------------------------------------------------- #
# Versies:  1.0.0   2014-09-01  Eerste versie.                                 #
################################################################################
# VERSION_NUMBER=1.0.5
# RELEASE_DATE=2016-12-28

Inleiding
=========
De scripts dienen tevens als voorbeeld ("showcase") van wat er mogelijk is
met bash shell scripting, zie ook Advanced Bash-Scripting Guide,
Mendel Cooper, http://tldp.org/guides.html/.

Voor uitleg over (het gebruik van) eCryptfs zie blog From the Canyon Edge
van Dustin Kirkland, http://blog.dustinkirkland.com en http://ecryptfs.org/.

Bronnen
=======
Voor gebruikte stijlen en standaarden zie:
1.  Linux kernel coding style
    http://www.kernel.org/doc/Documentation/CodingStyle
2.  Rob Pike's Notes on Programming in C
    http://doc.cat-v.org/bell_labs/pikestyle
3.  The Art of Unix Programming van Eric S. Raymond
    http://www.catb.org/esr/writings/taoup/html/
4.  Advanced Bash-Scripting Guide
    http://tldp.org/LDP/abs/html/abs-guide.html#SCRSTYLE
5.  BashGuide - Practices
    http://mywiki.wooledge.org/BashGuide/Practices
6.  The Bash-Hackers Wiki - Scripting with style
    http://wiki.bash-hackers.org/scripting/style

Voorts zijn er technieken gebruikt beschreven op de volgende websites:
http://robertmuth.blogspot.nl/2012/08/better-bash-scripting-in-15-minutes.html
http://www.linuxjournal.com/content/return-values-bash-functions

De scripts zijn gecontroleerd met checkbashisms uit pakket devscripts, en
via de site http://www.shellcheck.net/.

Opmaak
======
In de scripts wordt een "indentation" van 4 spaties gebruikt (geen TABs).
Maximaal gebruikte regellengte is 77.

Variabelen
==========
Als variabelen worden gebruikt:
scope (en)  scope (nl)  beschrijving
----------  ----------  ------------
local       functie     varnaam kleine letters, variabelen, syntax: local
<varnaam>
                        wordt gebruikt als functie lokale variabelen heeft
global      script      VARNAAM grote letters, constanten, syntax: readonly
<VARNAAM>
```

```
                          VARNAAM grote letters, variabelen, syntax: declare
    <VARNAAM>
environment omgeving/   VARNAAM grote letters, constanten en variabelen,
            shell en    syntax: export <VARNAAM> of declare -x <VARNAAM>
            subshell    export wordt *niet* gebruikt in mijn scripts

Code-block
==========
{
    <code>
}
Code-blok, "curly brackets", wordt gebruikt in mijn scripts.
Ook wel "inline group" genoemd. Deze constructie maakt, als het ware, een
anonieme functie (een functie zonder naam).
Echter, anders dan bij een "standaard" functie, blijven de variabelen in zo'n
code-blok zichtbaar voor de rest van het script.
(Ref. Advanced Bash-Scripting Guide)

If-(elif-)then-else constructies
================================
Logical compounds (logische samenstellingen):
[[ 1 -eq 1 ]] && echo 'waar'        (output: waar)
[[ 1 -eq 2 ]] || echo 'niet-waar'   (output: niet-waar)

Variatie #1 (eenvoudige selectie met één regel)
<opdracht> && return 0

<opdracht> kan ook een boolean zijn, de afsluitwaarde is bepalend

Variatie #2 (eenvoudige selectie met meerdere regels)
<opdracht> ||
    {
        <code>
        <code>
    }

Variatie #3 (complexe selecties met elif/else tak met meerdere regels):
if [[ 1 -eq 1 ]]; then
    Aaaa
    Bbbb
elif [[ 1 -eq 2 ]]; then
    Cccc
    Dddd
else
    Eeee
    Eeee
fi

Als <script> (ook) wordt gesourced (. <script> of source <script>)
==================================================================
Een 'readonly' vervangen door 'declare' vanwege de melding
"bash: <variabele>: is een alleen-lezen variabele".
Een 'exit 1' sluit het terminalvenster!
Vandaar als gesourced ($basename "$0") = 'bash') alleen return 1 (of 0) en
geen exit.
In een functie:
    bij een fout  - return 1:
        {
            error "Kan profiel $HOME/.profile niet inlezen."
            [[ $(basename "$0") = 'bash' ]] && return 1 || exit 1
        }
    einde functie - return 0 (nodig als een laatste opdracht 1 geeft maar dat
                               is OK, moet de functie toch met 0 eindingen):
    return 0
In de hoofdlijn na aanroep van een functie (nu opgelost in verwerk_rc_SC):
```

```
128       [[ $? -gt 0 ]] &&
129           {
130               [[ $(basename "$0") = 'bash' ]] && return 1 || exit 1
131           }
132   Bij het afsluiten:
133       [[ $(basename "$0") = 'bash' ]] && return 0 || exit 0
134   Voor $(basename "$0") kan evt. $PROGNAME gebruikt worden na het inlezen van
135   script-common.sh.
136   Zie de scripts getstarted en script-common.sh.
137
138   Opties en argumenten
139   ====================
140   [BESTAND...]         0 of meer BESTANDen
141   [ -f BESTAND...]     als optie f, 1 of meer BESTANDen, -f bestand1 bestand2
      etc.
142   [-f BESTAND]...      als optie f, 1 BESTAND, -f bestand1 -f bestand2 etc.
143
144   Einde tekstbestand.
```