
Bash Functies en Tellen



Functies

Tellen

Functies

Tellen

Bash functie

```
#!/bin/bash
```

```
# Functie: Declaratie van de function "functie"
```

```
#           Oproepen van functie
```

```
function functie(){  
    echo "Dit is de functie"  
}
```

```
functie
```

Bash functie eigenschappen

- ❑ Een functie MOET gedefinieerd zijn VOOR je ze gebruikt
- ❑ Een functie mag NIET leeg zijn
 - ▣ ook niet enkel commentaar
- ❑ De argumenten \$1, \$2,... zijn NIET de argumenten van het script maar van de functie

Bash functie met argument

```
#!/bin/bash
```

```
# Functie:    Tonen van het argument in kleur
```

```
#            kleuren van output
```

```
reset='[0m'    # Vergeet NIET te resetten of alles  
blijft gekleurd!
```

```
rood='[0;31m'
```

```
function tooninkleur() {  
    echo -e "\e$rood $1 \e$reset"  
}
```

```
tooninkleur "Hallo"
```

```
tooninkleur "Tamelijk rood"
```

Voorbeeld:
H4_kleur.sh

Bash functie met argument

- Wat extra info:
- The ANSI [\[1\]](#) escape sequences set screen attributes, such as bold text, and color of foreground and background.
- [DOS batch files](#) commonly used ANSI escape codes for *color* output, and so can Bash scripts.
- <http://tldp.org/LDP/abs/html/colorizing.html>
- http://misc.flogisoft.com/bash/tip_colors_and_formatting

Bash functions: return waarde

- Een functie kan in bash eigenlijk alleen maar **gelukt** of **niet gelukt** teruggeven

```
function checkexist(){  
    if [ -f "$1" ]; then  
        return 0  
    else  
        return 1  
    fi  
}
```

```
if checkexist "/etc/passwd" ; then echo passwd ok; fi
```


Bash functions: return

```
#!/bin/bash
```

```
# Functie: "Return" waarde van een functie is een  
string
```

```
function tooninkleur() {  
    local reset='[0m'  
    local rood='[0;31m'  
    echo -e "\e$rood $1 \e$reset"  
}
```

```
resultaat=$(tooninkleur "ok")  
# of resultaat=`tooninkleur "ok"`  
echo $resultaat
```

Functies

Tellen

Bash tellen

- `+` `-` `/` `*` `%` `**` (plus, min, delen, maal, mod, expon.)

- Met `expr`

- `a=1`
`b=2`
`som=`expr $a + $b``

- Met `let`

- `let "som=$a+$b"`

- In `bash`

- `som=$(($a + $b))`
`echo $som`

- ◆ **Opm: `echo $((3/2))` → geeft 1**
 - ◆ **Voor komma getallen gebruik je `"bc"`**

Voorbeeld:
`H4_obase_ibase.sh`

Bash tellen

Voorbeeld:
H4_obase_ibase.sh

BC

Simpel script:

Voorbeeld:
./H4_bc_roundup.sh

- `#!/bin/bash`
- `echo '6.5 / 2.7' | bc:`

Je kan bc parameters zetten:

- **Scale** = hoeveel getallen na komma
 - geen afronding (evt. Zelf berekenen → functie schrijven...)
- Conversies tussen talstelsels:
 - **Obase** = basis voor output
 - **Ibase** = basis voor input