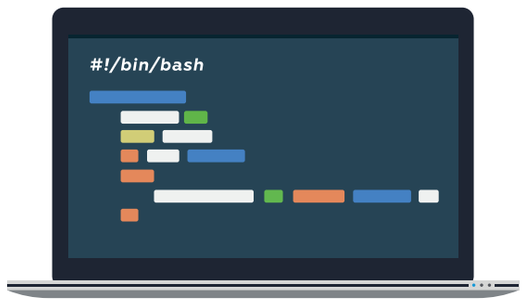
LISTING UITLEG

```
Start nieuwe regel
regel met spatie na afbreking
regel zonder spatie na afbreking
```

Meer informatie over de listing-uitleg vind je in de inhoudsopgave.



Shell scripting

DEEL 2: EENVOUDIG EN SNEL PROGRAMMEREN

In het eerste deel van deze workshop beschreven we de basis van shell scripting. In dit tweede deel van deze serie borduren we daarop voort. We gaan kijken naar enkele nieuwe aspecten, maar we diepen ook deel 1 een stukje verder uit. Met al deze kennis heb je dan voldoende bagage om de meest voorkomende problemen met eigen scripts op te lossen. **Serge Gielkens**

Het eerste deel van deze workshop verscheen eerder in Linux Magazine 3. De informatie uit dat artikel gebruiken we hier zonder verdere toelichting. Mocht je daarom iets tegenkomen dat je onbekend voorkomt, sla dan deel 1 er nog eens op na.

LOOPING

We beginnen met een klein script om twee woorden onder elkaar te tonen:

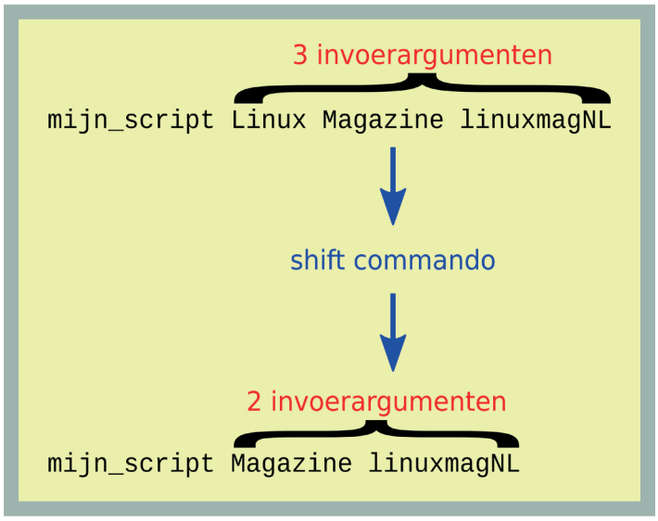
```
#!/bin/bash

woord1=$1
woord2=$2

echo $woord1
echo $woord2
```

De uitvoer ziet er nu als volgt uit:

```
./mijn_script Linux Magazine
Linux
Magazine
```

Het is makkelijk om het script voor drie (of een ander vast aantal) invoervariabelen aan te passen, maar hoe verwerk je nu een onbekend aantal invoervariabelen? Dat doe je met de while-loop:

```
while <test> ; do <commando's>
; done
```



Afbeelding 1: Invoervariabelen afhandelen

Dit is een samengesteld commando, net als de if-constructie uit deel 1. Merk hierbij ook de gelijkenis op qua syntax:

```
if <test> ; then <commando's>
; fi
```

De while-loop voert allereerst de test uit. Als die slaagt, gaat hij door met de commando's. Na afloop gaat hij weer terug naar de test en begint de cyclus opnieuw. Daarom moet je ervoor zorgen de commando's op enig moment de test doen falen. Zo niet, dan loopt je script eeuwig door.

Met de while-loop toon je een willekeurig aantal invoervariabelen als volgt:

```
while [ $# -ne 0 ] ; do
        woord=$1
        echo $woord
        shift
done
```

Het commando **shift** schuift de rij invoervariabelen een plaats naar links op (zie afbeelding 1). Daarbij valt de eerste variabele weg, zodat de tweede variabele nummer één wordt. Daarmee neemt het aantal invoervariabelen met één af en wordt het tenslotte 0. Dan faalt de test en stopt de loop.

Nadeel hierbij is dat je de variabele **woord** telkens een nieuwe waarde geeft. Je weet niet meer wat de invoer geweest is. Met een array kan dat gelukkig wel.

ARRAY

Een array is een lijst van elementen met een index. In Bash definieer je een array door de reeks elementen tussen haakjes te plaatsen:

```
mijn_array=(Linux Magazine)
```

Met een getal tussen vierkante haken kies je hieruit een element. Dat is je index. De nummering begint bij 0, dus het eerste element toon je door:

```
echo ${mijn_array[0]}
Linux
```

De accolades zijn verplicht, anders beschouwt de shell het woord **mijn_array** als variabelenaam met daarachter letterlijk **[0]**. Gebruik het apenstaartje om alle elementen in één keer te tonen:

```
echo ${mijn_array[@]}
Linux Magazine
```

Een nieuwe waarde geven of een element toevoegen, gaat als volgt:

```
mijn_array[2]=linuxmagNL
```

Bovenstaande while-loop hoef je maar een beetje aan te passen om de woorden in een array te plaatsen: