Korte uitleg: Environment variabelen (zoals PATH, DISPLAY, LANG, ...)

Zoals in *Korte uitleg: Processen* verteld, hoort bij elk proces (draaiend programma) een stukje geheugen met gegevens behorende bij dat process. Een deel daarvan is gevuld met Environment variables (omgevingsvariabelen). Iedere Environment variable heeft een naam en een tekstinhoud. Je kunt ze opvragen bij de Kernel en dan zijn ze dus goed leesbaar. `bash` (en andere shells) hebben daar een commando voor: `env`. Omdat dat nogal veel uitvoer is beperk ik dit hier even:

```
henk@boven:~> env | head
LESSKEY=/etc/lesskey.bin
XDG_VTNR=7
MANPATH=/usr/local/man:/usr/share/man:/opt/kde3/share/man
NNTPSERVER=news
SSH_AGENT_PID=2642
KDE_MULTIHEAD=false
XDG_SESSION_ID=1
DM_CONTROL=/var/run/xdmctl
HOSTNAME=boven
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
henk@boven:~>
```

Je kunt ook de inhoud van een environment variable op het scherm krijgen met:

```
henk@boven:~> echo $HOSTNAME
boven
henk@boven:~>
```

Hé, wacht even, maar als ik in `bash` werk is dat toch ook de manier om mijn eigen variablelen te gebruiken? Zoals in:

```
henk@boven:~> AAP=noot
henk@boven:~> echo "De inhoud van AAP is: $AAP"
De inhoud van AAP is: noot
henk@boven:~
```

Ja, en dat is verwarrend.

Als je de naam van een variabele gebruikt kijkt `bash` eerst of er een variable van die naam is "lokaal binnen de shell sessie". Als die er niet is wordt gekeken in de environment. Bij bovenstaand gebruik van `HOSTNAME` weten we alleen dat die geen "lokale" variable is omdat we daarboven hebben gezien dat hij in de environment zit. De variable `AAP` bestond nog niet (geloof me maar) en werd lokaal aangemaakt. We kunnen dus:

- variabelen uitlezen, lokaal en uit het environment (want er bestaat er

maar ééntje);
- variabelen wijzigen, lokaal en in het environment (want er bestaat er maar eentje);
- variabelen creëren (zoals AAP), maar alleen lokaal.

Hoe creëren we nu een nieuwe variable in de environment? Met het `export` statement verplaatsen we een variable van lokaal naar de environment:

```
henk@boven:~> env | grep AAP
henk@boven:~> export AAP
henk@boven:~> env | grep AAP
AAP=noot
henk@boven:~> echo "De inhoud van AAP is; $AAP"
De inhoud van AAP is; noot
henk@boven:~>
```

Zoals we zien staat `AAP` met zijn waarde nu in de environment. Je kunt dit overigens korter doen:

```
export NOOT=aap
```

zet `NOOT` meteen in de environment en geeft `NOOT` de waarde `aap`.

Uit het environment halen:

```
henk@boven:~> export -n AAP
henk@boven:~> env | grep AAP
henk@boven:~> echo "De inhoud van AAP is; $AAP"
De inhoud van AAP is; noot
henk@boven:~>
```

Let op! Sommige environment variables kunnen niet gewijzigd worden (read-only), lees de man pagina van `bash`.

Wat doen we er mee?

Environment variables zijn er vooral om variabele=waarde paren door te geven aan een proces. En een proces kan door toevoegen, wijzigen of weghalen weer waardes doorgeven aan een kindproces. Aangzien een proces een draaiend programma is kan de programmeur dus iets doen met die paren. Echter, die variable=waarde paren worden niet centraal geregistreerd. En dat betekent dat ieder programma dus zelf namen van variabelen kan kiezen. Dubbel bezettingen kunnen dus voorkomen. In de praktijk valt dit mee. Je kunt bijv. als programmeur jouw variabelen laten beginnen met de naam van je programma. De kans op dubbel gebruik wordt dan kleiner. Als het goed is staat natuurlijk in de gebruiksaanwijzing van een programma welke environmment variables gebruikt worden en welke waardes ze mogen hebben.

`bash`, gecompliceerd programma als het is, maakt een hele rij environment

varables waar je gebruik van kunt maken. De shell gebruikt ook een hele rij environment variables, die bij het starten van het `bash` proces kunnen worden meegegeven. Beide lijsten vindt je in de `man` pagina.

Hieronder een aantal variablen die gebruikt worden door veel programma's of die van belang kunnen zijn:

`TZ`

de time zone die gebruikt wordt om systeemtijden om te rekenen naar de datum/tijd die je ziet (en omgekeerd):

```
henk@boven:~> date
zo mei 18 21:01:16 CEST 2014
henk@boven:~> TZ=Asia/Kathmandu date
ma mei 19 00:46:18 NPT 2014
henk@boven:~>
```

en:

```
henk@boven:~/test> ls -l file
-rw-r--r-- 1 henk wij 21  8 jul 16:00 file
henk@boven:~/test> TZ=UTC ls -l file
-rw-r--r-- 1 henk wij 21  8 jul 14:00 file
henk@boven:~/test>
```

Je ziet hier ook hoe je een environment variable kunt meegeven in een commando, waarbij het alleen wordt meegegeven aan het te starten process (`date` in dit geval), maar geen invloed heeft op je huidige `bash` sessie.

`LANG`

gebruik de aangegeven taal voor eventuele meldingen.

```
henk@boven:~/test/wdir> l
totaal 8
drwxr-xr-x 2 henk wij 4096  8 jul 18:32 ./
drwxr-xr-x 4 henk wij 4096 10 jul 17:33 ../
henk@boven:~/test/wdir> LANG=C l
total 8
drwxr-xr-x 2 henk wij 4096 Jul  8 18:32 ./
drwxr-xr-x 4 henk wij 4096 Jul 10 17:33 ../
henk@boven:~/test/wdir>
```

Het verschil is hier minimaal: `totaal` vs. `total` en de manier van datum aangeven. `C` is hier hetzelfde als geen, en dus systeem engels.

`DISPLAY`

Een belangrijke variable gebruikt door GUI programma's. Dat wil zeggen, de programma's maken gebruik van library routines om windows te maken, te wijzigen en te verwijderen en deze libry routines gebruiken `DISPLAY` variabele om te weten op welke X-server dat gedaan moet worden. Op een PC met Linux is dat meestal:

```
henk@boven:~> echo $DISPLAY
:0
henk@boven:~>
```

en dat betekent scherm 0 op het lokale systeem. Maar als er meer ingelogde gebruikers zijn, zal die `0` bijv. `1` of `2` zijn. En ook vóór de `:` kan een ander systeem zijn vermeld (IP adres of hostnaam). Als er op de CLI is ingelogd (verwar dit niet met het gebruik van een terminal programma in de GUI, dan ben je in die GUI ingelogd, niet in de CLI) is `DISPLAY` helemaal niet gezet.

`HOME`

Krijgt als waarde het pad van de home directory van de gebruiker zoals geconfigureerd in `/etc/passwd`.

`PATH`

Bevat een lijst van directories waarin de shell gaat zoeken naar commando's waarvan geen pad is vermeld (en die niet builtin van de shell zijn). Zie *Korte uitleg: Welk bestand wordt uitgevoerd bij een commando?*.