Korte uitleg: Het Magische Getal Shebang en Bestandsnaam Extensies

Dat is een moeilijke titel en ik hoop dat hij niet afschrikt.

Even een korte inleiding over "Magic numbers" voor we overstappen op de "Shebang".

Een bestand bestaat uit nul of meer bytes, maar wat die bytes voorstellen weet bijna niemand zonder verdere uitleg. Als je weet dat ieder byte een ASCII teken vertegenwoordigt, kun je die tekens dus afdrukken en dan (hopelijk) de tekst lezen. Als telkens 4 bytes een groot heel getal vormen en twee opvolgende van die getallen samen een noorderbreedte en een westerlengte dan kun je een rij van dat soort punten op een kaart tekenen. Maar dat moet je wel weten.

We hebben dus gegevens nodig **over** het bestand (zgn. meta-data). Helaas, bij het begin van Unix is daar niet echt aan gedacht. Het enige wat gedaan kan worden is kijken in (het begin van) het bestand. Dan is er vaak een slimme gok te wagen. Soms omdat bepaalde bestandstypes van zichzelf daar iets speciaals neerzetten (zo beginnen GIF bestanden met de ASCII tekens `GIF87a` of `GIF89a`).

Het programma `file` is erg goed in het doen van deze slimme keuzes. Ga eens in een directory staan met allerlei soorten bestanden en doe:

```
henk@boven:~/test/bestanden> ls -l *
-rw-r--r-- 1 henk wij   3964 10 mei 20:42 ASH.html
-rw-r--r-- 1 henk wij     23 13 jul 17:32 directory
-rw-r--r-- 1 henk wij    796 10 mei 20:41 index.html
-rwxr-xr-x 1 henk wij 114424 13 jul 17:32 ls
-rw-r--r-- 1 henk wij 157943 10 mei 20:39 verf.gif
-rw-r--r-- 1 henk wij 157943 10 mei 20:38 verf.jpeg
-rw-r--r-- 1 henk wij    122 10 mei 20:40 WARNING_README.txt
henk@boven:~/test/bestanden> file *
ASH.html:           HTML document, UTF-8 Unicode text
directory:          ASCII text
index.html:         HTML document, ASCII text
ls:                 ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically li
verf.gif:           JPEG image data, JFIF standard 1.01
verf.jpeg:          JPEG image data, JFIF standard 1.01
WARNING_README.txt: ASCII text
henk@boven:~/test/bestanden>
```

(Er zijn hierboven wat eigenaardigheden, zoek ze op).

Sommigen zullen zeggen: maar daar hebben we toch extensies voor? Helaas, het Linux systeem weet niets van extensies. Deze file name extensions komen van MS-DOS en zijn een eigen leven gaan leiden. Ze hebben ook enkele nadelen:

- niets weerhoudt iemand er van om zo'n extensie te veranderen, er is dus geen zekerheid (zie `verf.gif` hierboven);
- er is geen centrale registratie en dus zijn veel combinaties meermalen in gebruik.

Sommigen zullen zeggen: maar het werkt toch op mijn systeem. Zoals gezegd weet het Linux systeem van niets. Iets anders is dat je in programma's/applicaties wel zoiets kunt inbouwen. En dat gebeurt ook. Soms heel strict, soms configurabel. Zo kun je in KDE configureren dat je bestandsnamen met een bepaalde extensie wilt laten openen door een bepaald programma als je ze aanklikt. Bijv. als de naam eindigt op `.gif`, open het dan met `gwenview`. De meesten zijn in KDE al voorgeconfigureerd.

Terug naar de Magic Numbers. Een script, dus een door een interpretator te interpreteren programma, moet aangeven welke interpretator gebruikt moet worden. Anders "weet" de Kernel dat niet. Voor een bash script is dat

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

Het teken `#` wordt in jargon vaak "hash" genoemd, het `!` "bang". De combinatie heet dan "shebang" of variaties daarop. Het is dus een onomatopee van geluiden geuit door nerds. Dit is dus ook een Magic Number. Merk ook op dat bij het uitvoeren door `bash` het `#` dit tot commentaar maakt, `bash` ziet dit dus niet.

Zonder shebang weet de Kernel dus niet of het een `python` of `csh` of `bash` of wat dan ook script is. Welliswaar kun je als gebruiker bij het aanroepen helpen door zelf die interpretator aan te roepen:

```
bash mijnscript
```

En zelfs direct aanroepen vanuit een shell sessie werkt omdat dan de shell van de sessie gebruikt wordt (bij gebrek aan betere informatie). Maar als je het script aanroep vanuit bijv. de `crontab`, dan wordt het een beetje moeilijk.

Ik zeg altijd: zonder shebang is een script geen script, maar een stelletje statements.

En als toetje: hoe maak je van een script een executable programma, net zo executable als binary executables (zoals bijv. `ls`, zie ook hierboven in het voorbeeld)?

1. Vergeet de juiste shebang niet.
2. Zet het script op een plek die in je `PATH` voorkomt (zie een apart verhaaltje, maar de directory `bin` in je eigen home directory is geen slechte plek).
3. Maak het executable voor de categorie(ën) naar keuze (alleen jijzelf, iedere gebruiker in je groep, iedereen) met het programma `chmod.`

Klaar. Je kunt nu je eigen script net zo aanroepen als `ls` of `file` of .....

```
mv mijnscript ~/bin/mijnscript
chmod u+x ~/bin/mijnscript
mijnscript
```