

Talstelsels

1. Inleiding

Wie professioneel met computers omgaat, krijgt te maken met verschillende talstelsels: tweetallig (*binair*), zestientallig (*hexadecimaal*) en soms ook achttallig (*octaal*). Dit verhaal legt enkele principes van talstelsels uit.

In het dagelijks leven gebruiken we “arabische” cijfers om getallen te bouwen. Maar soms gebruiken we ook andere technieken: Romeinse cijfers, turfjes zetten, of schuiven met kralen in een telraam (abacus). Wanneer we met een lege mand naast een berg appels staan, en we schriftelijk opdracht krijgen om een bepaalde hoeveelheid appels in de mand te doen, kan het werkbriefje geschreven zijn in arabische of Romeinse cijfers, maar er kan ook een hoeveelheid turfjes op staan, en het zou zelfs met een (tekening van een) telraam kunnen.

Het is belangrijk om de volgende twee zaken uit elkaar te houden:

- De hoeveelheid die we bedoelen (appels in de mand)
- De schrijfwijze die we gebruiken om de hoeveelheid aan te duiden.

Een bedoelde hoeveelheid “bestaat” onafhankelijk van de techniek die we kiezen om die hoeveelheid aan iemand anders duidelijk te maken.

De arabische cijfers zijn in Europa ingevoerd door kooplieden en bankiers in de 16e eeuw. Ze bleken veel handiger te werken dan de Romeinse cijfers. Hét grote idee bij die arabische cijfers is dat er een component bij zit om *nul* aan te geven. De Romeinen hadden niet bedacht dat je voor “niks” ook iets zou willen opschrijven. Maar de nul levert een handige techniek om de *plaats* (kolompositie) van een cijfer binnen een getal te laten meewerken aan de betekenis. Als iemand € 209.642,- op een bankrekening heeft staan, dan is de linker 2 véél interessanter dan de rechter. En de 0 in dit voorbeeld kun je niet zo maar schrappen, ook al stelt hij op zichzelf geen enkele Euro voor. Dit mechanisme heet een *positioneel talstelsel*.

2. Het tientallig stelsel

Het arabische mechanisme gebruikt 10 cijfers: er zijn tien verschillende symbooltjes ontworpen die we mogen gebruiken. Waarom zijn het er tien, en niet zeven of twaalf of... Waarschijnlijk komt dat omdat we tien vingers hebben, en we ons daarom al vanaf de prehistorie vertrouwd voelen met dat aantal. We noemen 10 het *grondtal* van ons talstelsel. Je hebt precies zóveel verschillende cijfersymbooltjes nodig.

Een getal stelt een hoeveelheid voor. Je kunt met een getal aangeven hoeveel appels je in de mand wilt hebben. De hoeveelheid wordt uitgedrukt door de gebruikte cijfers, én door hun kolompositie binnen het getal. Kijken we wat nauwkeuriger naar (bijvoorbeeld) het getal 7082:

```
7 0 8 2
      ^ twee eenheden
      ^ acht tientallen
      ^ nul honderdtallen
      ^ zeven duizendtallen
```

en dat allemaal opgeteld is de hoeveelheid die we bedoelen.... Nu formuleren we het wat technischer:

```
7 0 8 2
      ^ twee maal tien-tot-de-macht-nul
      ^ acht maal tien-tot-de-macht-een
      ^ nul maal tien-tot-de-macht-twee
      ^ zeven maal tien-tot-de-macht-drie
```

Dus: elk cijfer bouwt als volgt mee aan de bedoelde hoeveelheid: de basiswaarde van het cijfer, maal grondtal-tot-“een macht die bepaald wordt door de kolompositie”. De kolomposities tellen van rechts naar links, en die telling begint bij nul.

Je kunt de werking ook bekijken zoals de kilometerteller in een auto werkt. Je zet een hoeveelheid wielmpjes naast elkaar, en op de omtrek van elk wieltje schilder je de cijfersymbooltjes (alle tien) in de juiste volgorde. Dan monteer je de wieltjes achter een venstertje, in de beginstand met de nullen naar voren.

Als de kilometers beginnen te tellen, begint het *rechter* wieltje te draaien. Het draait totdat al zijn cijfers eenmaal voorbij het venster gekomen zijn (0 → 9) en na de 9 draait het gewoon verder door naar de 0. Dus na het hoogste cijfer komt het laagste cijfer weer aan de beurt. Maar: als effect van die hoogste→laagste doorgang wordt het wieltje dat links ernaast zit één stapje meegetrokken. Zo werkt het voor alle wieltjes in de rij.

3. Tweetallig stelsel

Computers werken onder de motorkap *binair*, tweetallig, met nulletjes en eentjes. In een computergeheugen wordt alles weergegeven met reeksen van nulletjes en eentjes: getallen waarmee moet worden gerekend, instructies die moeten worden uitgevoerd, randapparaten die moeten worden bediend, de adressen van alle computers aan het Internet, enz. enz.

Tweetallig wil zeggen: we gebruiken twee cijfers (0 en 1). Het betekent ook: we werken met *grondtal 2*. Zouden we een kilometerteller op die manier bouwen dan schilderen we op de omtrek van elk wielkje onze *twee* cijfers. Verder blijft de werking hetzelfde, al zullen we snel véél wieltjes naast elkaar nodig hebben. Tweetallige getallen zijn al gauw langer dan tientallige, en daardoor voor mensen veel onhandiger. Maar ze zijn precies even exact om een aantal kilometers weer te geven, of om appels in een mand te tellen.

Laten we eens de loop van een tientallig telwerk naast een tweetallig telwerk zichtbaar maken; we laten ze gelijk-op lopen:

<i>tientallig</i>	<i>tweetallig</i>	
00	00000	<i>de beginstand</i>
01	00001	
02	00010	
03	00011	
04	00100	
05	00101	
06	00110	
07	00111	
08	01000	
09	01001	
10	01010	
11	01011	
12	01100	
13	01101	
14	01110	
15	01111	
16	10000	
17	10001	<i>en zo verder...</i>

Als je beide kolommen nauwkeurig bekijkt, dan zie je de wieltjes op precies dezelfde manier functioneren: steeds als het hoogste cijfer op een wielkje gepasseerd is, begint het opnieuw bij het laagste, en trekt zijn linkerbuurman één stapje mee. Dat is onafhankelijk van het feit of er tien of twee cijfers op een omtrek staan.

Bestudeer ook eens aandachtig hoe op een verticale positie binnen de rechterkolom het ritme van nullen en enen elkaar afwisselt: helemaal rechts gaat het 010101010101, links daarnaast gaat het 001100110011, weer links daarnaast gaat het 0000111100001111, enzovoort.

De beide notaties zijn even exact en nauwkeurig. Maar ze zijn niet even handig in het gebruik: de linker methode is handiger voor mensen, de rechter is handiger voor computers.

Als oefening moet u maar eens uitrekenen welke hoeveelheid het volgende tweetallige getal voorstelt:

01001
^ ^ een maal twee-tot-de-macht-nul
^ een maal twee-tot-de-macht-drie

Realiseer u wel dat u nu een uitkomst in het tientallig stelsel aan het vormen bent. Maar dat is wel de manier om in de tabel te controleren of u goed gerekend hebt (tweetallig 01001 correspondeert met tientallig 9).

De twee cijfers 0/1 waarmee we op die manier werken heten *bits* (binary digits). Het talstelsel noemen we “tweetallig” of, in potjeslatijn *binair*. Voor de volledigheid: ons normale tientallig stelsel heet ook *decimaal*.

Een computergeheugen is opgedeeld in “woorden”. Een “woord” is eigenlijk de basisportie waarmee die computer het snelst overweg kan. Wanneer men spreekt over een 32-bits computer (alle PC’s met Intel processor zijn dat) dan bedoelt men dat een geheugenwoord bestaat uit 32 bits. Als we in de tellertabel van de vorige bladzijde de rechterkolom hadden laten doorlopen totdat er 32 eentjes naast elkaar stonden (dus het hoogste getal dat een 32-bits woord kan bevatten voordat het “de klok rond” gaat), dan had de linkerkolom op 4294967295 gestaan: 4 Giga, ofwel $2^{32}-1$.

De term *byte* staat (tegenwoordig) voor een portie van 8 bits. Ook dat is een portiegrootte waarmee een computer efficiënt uit de voeten kan. De term *nibble* staat voor een halve byte (4 bits).

In de zware servers kom je 64-bit processors tegen, en in de wereld van besturingsapparatuur (videorecorder, wasmachine) bestaan nog 8- en 16-bit processors. Nòg andere ontwerpen zijn zeldzaam, maar bestaan ook.

4. Hexadecimaal

Soms moet je als professional onder de motorkap van je computer duiken, en krijg je te maken met die bytes of woorden vol nullen en enen. Dat kan voorkomen als je troubleshooting moet doen, of als je een programma ontwerpt waarin bepaalde stukken duidelijker uitkomen als je ze “op de computermanier” opschrijft. Maar die lange reeksen blijven vreselijk onhandig, en als je ze moet opschrijven ook erg foutgevoelig.

Daarom kiest men een tussenoplossing: als we eigenlijk een tweetallig getal bedoelen, pakken we daarvan steeds porties van 4 bits (nibbles) bij elkaar. Met vier bits kun je 16 combinaties bouwen (dat kun je zien in de tabel hierboven, en je kunt het ook uitrekenen: $2^4 = 16$). Voor elk van die 16 combinaties verzinnen we één cijfersymbooltje, en dus krijgen we daarmee een 16-tallig talstelsel. In potjeslatijn heet dat *hexadecimaal*.

Maar er is een probleempje: een grafisch ontwerper zou 16 nieuwe symbooltjes voor ons moeten ontwerpen, en dat is te veel gevraagd. Dus we doen wat we ook al bij het tweetallig stelsel deden: leentjebuurt spelen bij de cijfersymbolen van het tientallig stelsel. Bij het tweetallig stelsel hadden we keus genoeg (we kozen 0 en 1), maar

voor een zestientallig mechaniek komen we nog zes symbooltjes te kort. Dan lenen we maar verder bij het alfabet, en zo krijgen we de cijferreeks 0 1 2 3 4 5 6 7 8 9 A B C D E F. Dat zijn nu onze 16 benodigde *cijfersymbolen* geworden.

We bouwen een nieuw kilometer-telwerk, en schilderen deze keer de 16 symbooltjes op de omtrek van elk wiel. Daarna laten we ook dat telwerk meelopen:

<i>tientallig</i>	<i>tweetallig</i>	<i>zestientallig</i>	
00	00000	00	<i>de beginstand</i>
01	00001	01	
02	00010	02	
03	00011	03	
04	00100	04	
05	00101	05	
06	00110	06	
07	00111	07	
08	01000	08	
09	01001	09	
10	01010	0A	
11	01011	0B	
12	01100	0C	
13	01101	0D	
14	01110	0E	
15	01111	0F	
16	10000	10	
17	10001	11	<i>en zo verder...</i>

Merk op hoe we op de hexadecimale teller in de linkerkolom een 1 krijgen op het moment dat de binaire teller helemaal door zijn meest rechtse portie van vier bits heengeteld heeft (bij de overgang van de twee-na-laatste naar de een-na-laatste regel).

Ook voor zestientallige getallen geldt weer de gebruikelijke methode als je wilt weten welke tientallige (decimale) waarde ermee correspondeert:

```
6BF4
  ^ vier          maal zestien-tot-de-macht-nul
  ^ F (=vijftien) maal zestien-tot-de-macht-een
  ^ B (=elf)      maal zestien-tot-de-macht-twee
  ^ zes           maal zestien-tot-de-macht-drie
```

Ter controle: hexadecimaal 6BF4 correspondeert met decimaal 27636.

5. Achttallig - octaal

Het UNIX systeem, en de programmeertaal C, zijn oorspronkelijk ontworpen op een computer uit de PDP-11 serie van het merk Digital Equipment. Die computer was ontworpen op een zodanige manier dat achttallig werken soms handig was. Dat betekent dus werken in een stelsel met acht cijfers: 0-7. Ook daarmee kun je natuurlijk weer een telwerk fabriceren, en verder is het van hetzelfde laken een pak. De potjeslatijn-naam voor achttallig is: *octaal*.

We demonstreren voor de laatste maal onze telwerken:

<i>tientallig</i>	<i>tweetailig</i>	<i>zestientallig</i>	<i>achttallig</i>
000	00000	000	000
001	00001	001	001
002	00010	002	002
003	00011	003	003
004	00100	004	004
005	00101	005	005
006	00110	006	006
007	00111	007	007
008	01000	008	010
009	01001	009	011
010	01010	00A	012
011	01011	00B	013
012	01100	00C	014
013	01101	00D	015
014	01110	00E	016
015	01111	00F	017
016	10000	010	020
017	10001	011	021

6. De programmeertaal C

In de broncode van uw programma moet u vaak getallen opschrijven. De compiler zal die uiteindelijk allemaal omvormen naar hun binaire vorm, en zo worden ze in het computergeheugen gezet.

Uiteraard is het voor een programmeur het gemakkelijkst als de compiler tientallig opgeschreven getallen accepteert. Maar soms is een programma-ontwerp gemakkelijker als sommige getallen hexadecimaal of octaal worden opgeschreven. Dat mag, als u maar duidelijk aangeeft dat de compiler ze dus moet bezien door een hexadecimale of octale bril. In de tellertabel hierboven maakt het echt verschil of je het getal 11 neemt uit de decimale kolom, uit de octale, of uit de hexadecimale.

De spelregels in de programmeertaal C zijn: een getal dat u opschrijft is in principe decimaal, tenzij het begint met een 0 (dan wordt het als octaal geteld) of het begint met 0x (dan wordt het hexadecimaal geteld). Let dus op dat u bij een willekeurig getal niet zonder meer wat nullen aan de linkerkant mag bijplakken. In het dagelijks leven mag dat wel, maar in C kan daardoor de betekenis veranderen.

7. En verder

Natuurlijk valt over talstelsels nog veel meer te vertellen. We hebben het niet gehad over negatieve getallen in een computer (2's complement notatie), getallen met een breukdeel (floating- en fixed point technieken), of over effecten die optreden in een berekening met getallen die een keiharde grootte-beperving hebben. We hebben ook niet gekeken naar getallen op een horloge, die "een wieltje verder draaien" na 12 of na 60 stapjes. Kortom: dit verhaal vertelt u enkele basisprincipes, maar u kunt zelfs bij computers nog veel meer talstelsel-techniek tegenkomen.