Cursusje LaTeX

M.A. Theijssen

voorjaar 2002

Proloog

Beste geïnteresseerde,

Je hebt interesse getoond voor een praktische cursus LaTeX, naast datgene wat je al hebt geleerd bij 'inleiding computergebruik voor mensen die al heel veel weten, maar datgene wat ze niet weten vertellen we niet'. Ik heb in 2002 een klein cursusje geschreven, daar is dit het eindresultaat van.

Ik er vanuit dat je je enigzins kunt redden op een X-terminal, zoals in de TK naast Marie of TK8 bij de drielinglift op de beganegrond; daarnaast ga ik er vanuit dat je op zijn minst Wilbert's fatsoenlijke instellingen hebt, in te stellen door

/vol/impuls/marcur/bin/ikwilfasoenlijkeinstellingen

in te typen in een windowtje en vervolgens uit en in te loggen. Dingen die je echt moet weten zijn de functies: mkdir, cd, ls, ls -l, rm.

Naast het praktisch gebruik van LaTeX wil ik ook iets zeggen over de manier waarop (La)TeX werkt, zodat je wat verder kunt kijken dan je neus lang is. Ook zul je dan (wel) waarderen waarom je als weldenkende bèta c.q. controlfreak beter af bent met LaTeX dan met een WYSIWYG-typesetter als Word.

Ik begin desalniettemin bij het begin en vertel hoe je een tex-bestandje maakt en wat je ermee moet doen om het te kunnen bekijken, printen en dat soort praktische dingen. Vervolgens komen dingen aan bod als het invoegen van figuren, het gebruik van input-bestanden, het typesetten van formules, het (her)definiëren van functies en nog veel meer.

Bij elke aflevering dient het bronbestand tevens als voorbeeld-bestandje waarin je dingen op kan zoeken. Ook kan ik je aanraden om *The Not So Short Introduction To LaTeX2e* uit te printen.

Veel plezier!

Marnix Theijssen

Inhoudsopgave

Hoofdstuk 1

My first TEX-file

1.1 Editors

Bij het maken van verslagen, brieven of andere teksten maak je in het algemeen gebruik van een tekstverwerker. Dat kan een hele simpele zijn, bijvoorbeeld een typmachine, maar ook een erg ingewikkelde met allerlei functies, zoals Word.

Wat tekstverwerkers allemaal doen is een bepaalde input, de op het toetsenbord ingevoerde tekst, vertalen naar een vel papier. Een typmachine doet dit mechanisch. Een elektronische, zoals Nedit, waar ik dit in schrijf, doet dit door de gegevens eerst op een beeldscherm te tonen en vervolgens het geheel af te drukken op een vel papier. Je zou genoegen kunnen nemen met de wijze waarop een typmachine tekst op het vel zet en de computer iets soortgelijks laten produceren. We weten echter allemaal dat je veel meer kunt dan droog letters drukken. Je kunt tekst vet drukken, schuin drukken, onderstrepen, maar ook op andere plaatsen zetten dan onder elkaar of groter of kleiner afdrukken. Dan zijn er ook nog zaken als voetnoten, titels, inhoudsopgaves en noem maar op.

Waar het nu om gaat is dat je het plaatsen van letters e.d., ook wel typesetting genoemd, op een dusdanige manier automatiseert, dat het op een makkelijke manier een mooi resultaat levert. Er zijn van deze automatiseringen grofweg twee soorten: WYSIWYG (what you see is wat you get) en WYPIWYG (what you program is what you get).

Een WYSIWYG-typesetter, zoals Word, bestaat uit drie componenten die samen een geheel vormen. De eerste component is een editor en die stelt je in staat om überhaupt tekst in te typen; direct na het intypen wordt de tekst getypeset en op het scherm getoond. Dit is de tweede component: de viewer. Als je daarna gaat printen wordt de beeldtaal omgezet naar papiertaal (en dus wordt de input nogmaals behandeld) en wordt het naar de printer gestuurd. Dit is de derde component: de printer. Het geheel kun je opslaan en met het opslaan wordt alle informatie wat betreft inhoud en opmaak in een bestand opgeslagen (en wel in zo'n formaat dat je hem alleen kan openen met hetzelfde programma).

LATEX heeft een andere insteek. Om de tekst in te kunnen typen gebruik je een (kleine) algemene editor, zoals NotePad of de goede oude DOS-editor. De tekst die je typt bevat (een deel van) de informatie voor de opmaak. Dit houdt in dat wanneer je een stuk tekst schuin gedrukt wil hebben, dat je daarvoor

commando's in je brontekst hebt staan: het woordje `\emph{tekstverwerken}` wordt zo schuin gedrukt. Vervolgens ga je de input typesetten, wat lijkt op een compiler voor bijvoorbeeld C, en je krijgt een bestandje (bij LATEX is dat een .dvi bestand) wat je bijvoorbeeld kunt viewen of printen. Je bronbestand blijft echter een tekstbestand en is dus met elke editor te wijzigen. De compiler doet dus eigenlijk niets anders dan het bronbestand van begin tot eind doorlezen en elke keer als er een commando staat iets aan de manier van typesetten veranderen.

1.2 Het kiezen van een editor

Een TEX-bestand (een bestand met de extensie .tex, wat aangeeft dat het geschikt is voor de LATEX-compiler) schrijf je zoals gezegd met een teksteditor. Daar is natuurlijk ruime keus in. Enkele editors die je op het netwerk van de faculteit kunt vinden zijn:

- VIM;
  een editor die erg lastig is om te leren maar erg snel werkt als je het eenmaal kan, bijkomend voordeel is dat het gewoon werkt in een xterm-window (vooral erg handig als je met PuTTY ofzo vanaf buiten inlogt en maar 1 windowtje hebt). Als je dit wilt leren, zul je moeten zoeken naar iemand ie het wel kan (of de manual pages doorploegen) want ik kan het niet.

- Nedit;
  een editor met een hoog windowsgehalt;, werkt op de terminals in TK1 wat trager dan de rest maar er is erg goed mee te werken.

- pico;
  de core editor die Pine gebruikt om mail op te maken. Hij ziet er ook uit als een Pine maileditor en dat is hij dan ook. Deze editor werkt ook in een window en is dus ook te gebruiken als je met PuTTY inlogt en windows openen te traag voor je is).

Ik zou je aanraden om Nedit te gebruiken, maar je moet het natuurlijk helemaal zelf weten. Het is natuurlijk ook heel goed mogelijk om een windows editor te gebruiken: een tekstbestand is immers een tekstbestand[1].

1.3 Een TEX-bestand schrijven

Laten we eens proberen om een klein TEX-bestandje te maken. Het is verstandig om voor TEX-bestanden een apparte directory aan te maken zodat je straks door de bomen het bos nog ziet. Ik zal aannemen dat je in je homedirectory een directory 'texfiles' hebt, zodat ik makkelijk kan verwijzen naar die bestandjes. Het bestandje dat we nu gaan maken noemen we: 'hello.tex'. Daarvoor open je in de dir 'texfiles' de editor, bijvoorbeeld met het commando:

```
nedit hello.tex &
```

[1] Voor mensen met afkickverschijnselen, heimwee of een andere smoes is het zelfs mogelijk om een TeX-bestand met Word te maken; bedenk wel dat je het moet opslaan als tekstbestand.

waarbij de & er voor zorgt dat je prompt meteen weer beschikbaar wordt. Wanneer het bestand hello.tex nog niet bestaat zul je in een dialoogvenster even moeten aangeven dat het een nieuw bestand betreft.

Nu gaan we eerst definiëren wat we eigenlijk gaan doen. Het moge duidelijk zijn dat een brief anders wordt opgemaakt dan een artikel en een artikel weer anders dan een boek. We gaan de compiler dus eerst vertellen met wat voor klasse document hij te maken heeft. Dit is altijd de eerste regel van een TEX-bestand:

```
\documentclass{article}
```

Het valt je misschien op dat deze code erg lijkt op algemene programmeertaal. En dat is ook niet zo verwonderlijk, want eigenlijk doe je precies hetzelfde. Je gebruikt echter de backslash voor een functienaam om aan te geven dat je te maken hebt met een functie. Een editor als nedit maakt hier overigens gebruik van en aan hand de extensie van het bestand (.tex dus) leest hij af dat hij woorden die met een \ beginnen vet moet drukken (bij een .html bestand doet hij vergelijkbare dingen).

Dus een commando begint altijd met een '\'. Tussen accolades '{}' staat altijd het argument van de functie. In het bovenstaande geval dus 'article' als argument van de functie `\documentclass`. Het is bij sommige functies mogelijk om naast een argument ook parameters bij het argument te geven. In bovenstaand geval bijvoorbeeld dat je niet de standaardgrootte van de letters wil gebruiken (10pt) maar een stukje groter (bijv. 11pt). Dit doe je met blokhaken voor het argument: `\documentclass[11pt]{article}`. Bij ons eerste TEX-bestand zullen we deze optie weglaten. Naast de documentclass 'article' zijn er natuurlijk nog meer. Voor deze cursus gebruik ik bijvoorbeeld de class 'report'. Daarnaast bestaan nog o.a. 'book' en voor de Impuls heeft Chris Dams ooit de documentclass 'impuls' gemaakt en idem voor de class 'smoelenboek'.

Nu de compiler weet wat hij voor zijn kiezen krijgt, kan hij beginnen met het verwerken van de eigenlijke tekst:

```
\begin{document}
```

Het gedeelte tussen `\documentclass` en `\begin{document}` kan gebruikt worden om nog meer zaken te definiëren, te herdefiniëren, in te stellen of op te geven. Het wordt de *preamble* van het document genoemd en het bevat dus alle informatie die vooraf bij de compiler bekend moeten zijn.

Nu kunnen we de tekst gaan schrijven, bijvoorbeeld het standaard zinnetje voor je eerste programma in een nieuwe taal:

```
Hello world!
```

Deze tekst komt dus straks linksbovenaan op het (eerste) vel papier te staan. Papierformaat, lettertype, marges en dergelijke worden automatisch met de documentclass geladen. Wanneer je hier veranderingen in wil maken, zul je dat in de preamble (of bij de opties van de documentclass) moeten doen. We komen hier later op terug.
Nu sluiten we het document af met

```
\end{document}
```

Je hebt nu het meest elementaire tex-bestand gemaakt. Als het goed is ziet het er zo uit:

```
\documentclass{article}
\begin{document}
hello world!
\end{document}
```

Een tweetal commando's `\begin{}` en `\end{}` vormen samen een 'environment'; een 'document' is een van de dingen die je kunt beginnen en eindigen, een vergelijking 'equation' is een andere (maar daar komen we nog op). Let op: elke `\begin{}` moet afgesloten door een dito `\end{}`! Je moet daar heel zorgvuldig in zijn, net zoals je dat moet met if-statements in bijvoorbeeld C.

Je programmaatje is nu af en klaar om op te worden geslagen! Ik stel voor om het bestand 'hello.tex' te noemen en het in je dir 'texfiles' te zetten.

1.4 Het compilen van een TEX-bestand

Nu we een tex-bestand hebben, gaan we het compilen zodat het een mooi leesen printbaar bestand wordt. Ga in de directory staan waar het bestand zich bevindt en typ

```
latex hello.tex
```

Je start nu LATEX op dat TEX aanroept (het typesetprotocol heet TEX, de compiler LATEX) en die zet de tekst neer op een manier zoals je dat hebt aangegeven in je TEX-bestand. Je ziet iets zoals:

```
This is TeX, Version 3.14159 (Web2C 7.3.2x)
(./hello.tex
LateX2e <1999/12/01> patch level 1
Babel <v3.6Z> and hyphenation patterns for english,
french, german, ngerman, dutch, dumylang,
nohyphenation, loaded.
(/vol/texlive5c/texmf/tex/latex/base/article.cls
Document Class: article 1999/09/10 v1.4a Standard
LaTeX document class
(/vol/texlive5c/texmf/tex/latex/base/size10.clo))
(./hello.aux) [1] (./hello.aux) )
Output written on hello.dvi (1 page, 232 bytes).
Transcript written on hello.log.
```

(Inderdaad: de versie van TEX gaat als pi!) En je ziet dat er een bestand hello.dvi gemaakt is! Tevens is er een auxillary-bestand aangemaakt (hello.aux) maar daar staat niets nuttigs in. De voortgang van de transcriptie wordt in een logfile (hello.log) geschreven. Deze logfile is handig om fouten op te sporen. Wanneer je een inhoudsopgave (table of contents) hebt gemaakt, wordt de informatie daarvan in een .toc-file opgeslagen.

Wanneer je een fout hebt zitten in je tex-bestand stopt de compiler met compileren en vraagt om instructies. Je krijgt een soort van prompt die voorafgegaan wordt door een vraagteken:

```
! LaTeX Error: \verb ended by end of line.

See the LaTeX manual or LaTeX Companion for explanation.
Type H <return> for immediate help.
 ...

l.149 ...ten: \\ \noi \verb|De verstandige student

? _
```

Je ziet dat er aangegeven wordt wat de fout is en bovendien in welke regel de fout te vinden is (preciezer: zijn oorsprong heeft). In dit geval staat er een verbatim-commando in regel 149 van het betreffende inputbestand dat niet goed werd afgesloten.

Je kunt nu een aantal dingen doen. Eén is `h` gevolgd door return wat, zoals aangegeven, een helpboodschap geeft (indien voorhanden). Andere mogelijkheden zijn 'return' wat de compiler vertelt 'gewoon' door te gaan; `s`, wat de scroll-mode induceert (alle foutmeldingen worden automatisch behandeld als met return); `x` wat het compilen direct onderbreekt zonder verder iets toe te voegen aan het outputbestand; `q` die het compilen verder als 'batchproces' uitvoert (er wordt geen output gegeven op het scherm).

1.5 Het dvi-bestand

dvi-Bestanden bekijk je met xdvi. Ga in de dir staan waar het dvi-bestand zich bevindt en typ

```
xdvi hello.dvi &
```

Er verschijnt een window met daarin de beeldschermvertaling van het dvi-bestand; de '&' zorgt er weer voor dat de promt waar je het commando gegeven hebt meteen weer beschikbaar is voor nieuwe input.

Je kunt nu bijvoorbeeld drie windows gebruiken om snel wijzingen te maken. De eerste bevat de editor met het TEX-bestand. De tweede een xterm waarin je je LATEX kunt draaien. De derde is het venster met xdvi waar de output te zien is. Een wijziging die je dan maakt in het .tex-bestand moet je bewaren, compilen en (met de knop 'reread') opnieuw laden in xdvi. Na enige oefening gaat dit sneller dan je in eerste instantie zou denken, zeker wanneer je op een beetje snelle machine werkt.

Een dvi bestand kun je ook uitprinten. Dit doe je door het commando

```
lpr -Plsrstud hello.dvi
```

in te typen in de directory waar het bestand staat. Je moet natuurlijk wel printbudget hebben. Je ziet dat 'lpr' het printcommando oproept met een optie om op de printer lsrstud te printen. Dit is de printer (alleen zwart-wit) die staat in de gang bij Marie naast de windows-tk. Op het schutvel dat bij elke printopdracht wordt afgedrukt vindt je alle mogelijke opties terug.

Wanneer er plaatjes in je bestand zijn ingevoegd is het noodzakelijk de uitvoer hierop af te stemmen. Je voegt namelijk alleen encapsulated-postsciptplaatjes in, wat op zichzelf staande stukjes printertaal zijn. Dit is niet helemaal compatibel met de rest van het dvi-bestand (dit zie je soms al als je het dvibestand bekijkt, met de v-toets onderdruk je het lezen van eps'jes in xdvi) en daarom moet je van het hele bestand een postscript maken; en dat doe je met dvips:

```
dvips -o hello.ps hello.ps
```

levert je je een bestand hello.ps (of iedere andere naam die je na de optie -o, van 'output', ingevuld hebt) dat je met gv kan bekijken en uitprinten (printen in gv geeft je een opdrachtregel met 'lpr'). Direct printen met lpr kan natuurlijk ook. Andere handige programma'tjes die bestanden in elkaar overvoeren zijn dvi2pdf (maakt van een dvi bestand een pdf, te bekijken met acrobat reader en dus een formaat wat geschikt is voor windowsbakken), ps2pdf (maakt van een ps een pdf) en psalm (wat op een erg handige manier je in staat stelt meerdere pagina's op een pagina te zetten, boekjes te maken, etc).

Hoofdstuk 2

Packages & Typografie

In dit deel leer je van alles m.b.t. het laden van speciale faciliteiten, de zogenaamde packages. Deze voegen allerlei handige functies toe op taalkundig, opmaaktechnisch en wiskundig gebied.

Tevens leer je hoe je je tekst zó moet typen dat je ook in staat bent allerlei speciale tekens te gebruiken, zoals koppeltekens, aanhalingstekens, accenten etc.

2.1 Packages

Wanneer je een verslag gaat maken wil je vaak allerlei extra dingen toevoegen. Een saai stukje platte tekst volstaat niet. Maar voor het ene verslag heb je andere dingen nodig dan voor het andere. Je kunt je bijvoorbeeld voorstellen dat het een verschil maakt of je bezig bent een verhaal te schrijven of dat je bijvoorbeeld een scriptie schrijft waar allerlei moeilijke formules in voorkomen. Ook schrijf je soms teksten in het Engels, soms in het Nederlands of voor mijn part in het Russisch.

Om nu te zorgen dat je niet allerlei overbodige fucties laadt, waardoor je bestand onnodig groot wordt, moet je in LaTeX afzonderlijke packages[1] laden. Een package laad je met behulp van het commando:

```
\usepackage[parameters]{packagename}
```

2.1.1 Taalkundig

Voor deze cursus laad ik o.a. het package babel. Hierdoor neem ik bijvoorbeeld afbreekroutines op, opdat LaTeX netjes voor mij woorden afbreekt tijdens het opmaken. Dit is een voorbeeld van een package dat invloed heeft op de manier waarop LaTeX omspringt met de taal tijdens het opmaken.

Wees gerust. Dit soort algemene packages doen geen rare dingen, zoals hoofd- in kleine letters veranderen, spelling automatisch corrigeren[2] of allerlei andere zaken.

Als je al naar de file cursus.tex gekeken hebt, dan is het je misschien al opgevallen dat ik door middel van de parameter dutch, tussen blokhaken natuurlijk,

[1] Dit zijn dus van te voren samengestelde bestanden waar specifieke functies in staan.
[2] want jij als schrijver weet natuurlijk zelf het beste hoe je iets vaut moet spellen

aangeef dat ik de afbreekroutines van het Nederlands wil gebruiken. Nu moet ik er volledigheidshalve bij vermelden dat LaTeX, in tegenstelling tot bijvoorbeeld Word, bij het uitvullen van regels uitgaat van alinea's en niet van regels op zich. Hierdoor is het afbreken van woorden bijna nooit nodig. Op deze pagina gebeurt het bijvoorbeeld maar vijf keer, terwijl er geen opvallende stukken 'wit' in de tekst te zien zijn.

2.1.2 Opmaaktechnisch

Wanneer je speciale dingen wilt met de opmaak, kun je hiervoor ook packages laden. Bijvoorbeeld wanneer je een index wil maken dan kun je het package makeidx gebruiken. Het package stelt je dan in staat om een index te maken. Een ander veelgebruikt package is het package `fancyhdr`. Dit package stelt je in staat om zelf kop- en voetteksten te definieren. Verderop staat een korte uitleg over het gebruik van dit package.

Waar het om gaat is dat je eraan moet denken dat speciale dingen op opmaaktechnisch gebied in sommige gevallen een extra package vereisen; wanneer je een functie opzoekt in bijvoorbeeld *The Not So Short Introduction To LaTeX2e* staat het er altijd bij vermeld of je een package moet laden.

2.1.3 Wiskundig

Een ander verhaal is het opmaken van wiskundige formules. Hoewel er standaard veel geregeld is in LaTeX, zul je vooral wanneer je speciale symbolen wilt gebruiken extra packages moeten laden. Voorbeelden zijn het package latexsym voor o.a. $\sqsubset, \bowtie \rhd, \rightsquigarrow$ en de packages eucal, eufrak en amsfonts voor verschillende fonts uit de wiskunde. (Denk hierbij bijvoorbeeld aan uitdrukkingen zoals '$\forall x \in \mathbb{R} : x^2 \geq 0$' waar de hoofletter R in dit lettertype het lichaam der reële getallen aangeeft. Om dit font te kunnen gebruiken, heb ik het package amsfonts (of amssymb) moeten laden.)

In de praktijk kom je er vanzelf achter wanneer je een extra package moet laden. Wanneer je namelijk opzoekt in *The Not So Short Introduction To LaTeX2e* welk commando je moet gebruiken voor een bepaald teken (bijvoorbeeld `$\sqsubset$` voor $\sqsubset$) dan staat er bij vermeld of, en zo ja welk package je moet laden. Het opmaken van wiskundige formules in detail komt nog uitgebreid aan bod.

2.2 Typografie

Naast de standaard tekens die op het toetsenbord staan, kent de (Nederlandse) taal nog een heleboel andere tekens. Deze tekens kun je met LaTeX natuurlijk ook gebruiken. Denk bijvoorbeeld maar aan accenten op letters. In andere talen komen zelfs tekens voor die je in het Nederlands nooit gebruikt zoals ¡, ¿, œ etc. Verder zijn er zaken als aanhalingstekens, koppeltekens en zo nog wat dingen. De manier waarop LaTeX hiermee omgaat is niet moeilijk, je moet het gewoon een keer gezien hebben.

2.2.1 Fonts en Speciale tekens

Om een speciaal teken in te voeren in bijvoorbeeld Word, moet je gaan naar een dialoogvenster 'Speciale Tekens' en vervolgens daar in een tabel een teken selecteren en inporteren in je document. De tabel waar je dat teken uit selecteert is eigenlijk niets anders dan de tabel van de karakters in het huidig gebruikte font[3]. In LaTeX is het eigenlijk weinig anders. Echter nu selecteer je niet rechtstreeks een teken uit de tabel maar je voegt het toe door een speciale (maar wel logische) letter- en tekencombinatie.

accenten

Een accent geef je gewoonlijk weer door een \ gevolgd door een teken dat het accent representeert, gevolgd door de letter waar het accent op moet.

`\'a`	á	`\~n`	ñ
`\‘a`	à	`\u a`	ă
`\^a`	â	`\'\i`	í

Speciale aandacht dient gegeven te worden aan het zetten van accenten op i'tjes en j'tjes. Wanneer je een accent op een letter wilt zetten, vertel je TeX eigenlijk dat hij eerst één karakter moet plaatsen (het accent) en vervolgens een ander daaronder (de letter). Nu moet je onder het accent niet een gewone i en j zetten maar een i of j zonder puntje, anders komt het accent en het puntje boven elkaar heen te staan; j̈ is niet de bedoeling.

Dat zit zo. TeX 'denkt niet na' over karakters en letters, het plaatst alleen balkjes met een bepaalde afmeting op een bepaalde plaats. Een accent is balkje en de letter waar het accent op moet is het ook. TeX doet niets anders dan die twee balkjes op een leuke manier boven elkaar zetten. Later worden die balkjes ingevuld. Het is dan ook logisch dat je een i zonder puntjes moet gebruiken.

Dit doe je door het commando `\i` te gebruiken, dit ziet er zo uit:

```
\'\i
```

In de nieuwere versie van TeX is het gebruik van accenten op de i enigzins vereenvoudigd. Je kunt namelijk nu ook gewoon `\'i` etc. gebruiken. Voor de j blijft bovenstaande echter nog steeds van kracht; maar goed hoe vaak zet je nu een accent op een j?

Wanneer je een i met een accent midden in een woord gebruikt, bijvoorbeeld het woord geïnteresseerd, dien je er rekening mee te houden dat je een commando moet afsluiten. In LaTeX doe je dat door middel van een spatie[4]. Het woord geïnteresseerd ziet er dus uit als `ge\"\i nteresseerd`; met een spatie tussen het commando `\i` en de `n`. De spatie zorgt ervoor dat het einde van het commando `\i` herkend wordt.

Nu maakt het in LaTeX niet uit of je nu één spatie of een heleboel spaties gebruikt om woorden te scheiden in je tex-bestand. Of ik nu `Er was eens` of `Er was  eens` als brontekst gebruik in beide gevallen levert het Er was

[3] Voor de word-specialisten: het dialoogvenster 'Speciale Tekens' opent standaard met een speciaal "speciale tekens"-font; dit doet echter niets af aan de essentie van het verhaal.

[4] Immers een woord, en dus ook een commando, loopt door tot aan een spatie.

eens[5]. Dit maakt het makkelijker om je brontekst overzichtelijk te houden maar wanneer je na een commando een spatie wil hebben, volstaat het niet om met één spatie het commando af te sluiten en met een volgende twee woorden te scheiden.

Een goed voorbeeld is het commando `\LaTeX` dat het woord LaTeX op een leuke manier schrijft: LaTeX. Wanneer je dit midden in een zin gebruikt 'De verstandige student zal al vrij snel LaTeX gebruiken', dan moet je aangeven dat het commando `\LaTeX` eindigt bij de X. Dit doe je door een extra backslash achter het commando te zetten:

```
De verstandige student zal al vrij snel \LaTeX\ gebruiken.
```

Op deze manier voorkom je dat je commando en het volgende woord aan elkaar geplakt worden. Een commando wordt ook afgesloten wanneer er een leesteken op volgt (een punt, komma, etc.). Bijvoorbeeld in

```
De slimme student gebruikt \LaTeX, dat spreekt voor zich.
```

Voor een completer overzicht van allerlei accenten verwijs ik naar de niet zo heel erg korte handleiding.

Het zetten van trema's wordt vereenvoudigd wanneer je gebruik maakt van het package babel. Het enige wat je hoeft te doen is een dubbel aanhalingsteken voor de letter te zetten: `"e, "i` levert 'ë, ï'.

In de Nederlandse taal dient het trema om lettergrepen te scheiden en het is daarom altijd een plaats waar een woord afgebroken kan worden. Nu is het zo dat wanneer er een woord afgebroken moet worden bij het trema, het trema vervalt. De package babel houdt hier rekening mee en zorgt dat dit automatisch goedkomt.
[voor de aardigheid: Hierboven heb ik het voorbeeld geïnteresseerd gebruikt om te laten zien hoe je een trema op een i moet zetten. Bij het maken van de tex-file voor deze cursus kwam het meerdere malen voor dat het woord geïntereseerd precies na de ge werd afgebroken. Uiteindelijk heb ik dit expliciet verboden door de e en de i met trema in één balkje te zetten. Dat doe je met `\hbox{}: g\hbox{e"i}nteresseerd`.]

leestekens

Naast letters en andere karakters bestaan er in de taal ook leestekens. Ik denk hierbij bijvoorbeeld aan punten, komma's, aanhalingstekens e.d. Hoe je hiermee omspringt hangt eigenlijk vooral af van je eigen schrijfstijl. Sommige mensen gebruiken veel korte zinnen andere brijen enorme lange monsters met puntkomma's, dubbele punten etc. Ik zal dan ook niet proberen te vertellen hoe je die dingen moet gebruiken maar alleen aangeven wat er 'te koop is'.

Allereerst de punten en komma's. Dit spreekt eigenlijk voor zich. Wanneer je de tekst schrijft laat je automatisch een spatie tussen de punt/komma en het volgende woord. Dat is dan ook precies wat je moet doen.

Deze spatie is voor LaTeX een variabele spatie. Dit houdt in dat hij de ruimte kan gebruiken om de regel op te vullen of om op een nieuwe regel verder te gaan. Dit is soms niet wat je wil. Een voorbeeld is de spatie tussen een titel/voorletters van een persoon en zijn achternaam: drs. D. van den Bergen. Het is niet mooi

[5] Iets dergelijks geldt ook voor alinea scheiding: alineas worden gescheiden door een open regel. Hier geldt dat één open regel hetzelfde behandeld wordt als twee of meer open regels.

wanneer er tussen de drs. en de D. veel ruimte zit of dat er een nieuwe regel begint. Dit kun je voorkomen door in plaats van een spatie een ~ te gebruiken: `drs.~D.~van den Bergen`. Wanneer een persoon meerdere titels heeft is het mooier om tussen de titels minder ruimte te laten. Dit kan met `\,`, wat een halve spatie is. Prof. dr. ir. J.C. Maan schrijf je dus als `Prof.\,dr.\,ir.~J.C.~Maan`.

Ook het gebruik van afkortingen vraagt enige voorzichtigheid. Een afkorting met puntjes midden in een zin laat die zin niet eindigen (je kent dat misschien van een slecht ingestelde Word, die automatisch hoofdletters gaat invoegen). In LaTeX moet je hier rekening mee houden, omdat de ruimte tussen een punt en het begin van de volgende zin namelijk extra gevoelig is voor vergroting bij het uitlijnen van de tekst. Dit kan verkeerd uitvallen, m.a.w. het kan erg lelijk zijn. Om dit te voorkomen moet je de afkorting eindigen met een \ d.w.z. dat je je schrijft `d.w.z.\`.

Een ander leesteken wordt gevormd door de drie puntjes die aangeven dat een verhaal verder... In bijvoorbeeld Word gebruik je gewoon drie puntjes, maar soms zie je dat hij ze op een of andere manier gaat groeperen. Wat Word probeert, is om ellipsen in te voeren, drie puntjes die samen een geheel vormen. De puntjes van een ellips staan wat verder uit elkaar. Vergelijk ... (drie puntjes) en … (ellips). Het commando voor de ellips is `\ldots`.

De l van het commando `\ldots` doet vermoeden dat er ook iets bestaat als `\cdots` en inderdaad ⋯ bestaat (het maakt echter onderdeel uit van de mathmode van LaTeX en dient daarom tussen $'s te staan, die de mathmode in en uitschakelen: `$\cdots$`). `\hdots` bestaat niet.

Het gebruik van aanhalingstekens is ook erg vanzelfsprekend. Je opent met ‘ of “ wat er in brontekst uitziet als `‘` en ` ‘‘`. Sluiten doe je met ’ of ” (in brontekst: `’` en `’’`). Wanneer je enkele aanhalingstekens gebruikt of dubbele, hangt eigenlijk van je eigen smaak af. Een wijdverbreid gebruik is dat in het Nederlands citaten met dubbele aanhalingstekens worden aangegeven en alle andere ‘dingen’ met enkele. In het Engels is het precies andersom.

2.2.2 Ligaturen

Het laatste waar ik het in dit hoofdstuk over wil hebben zijn ligaturen. Ligaturen zijn combinaties van letters die niet worden weergegeven door de letters afzonderlijk achter elkaar te zetten, maar door een specifiek karakter te gebruiken. Dit gebeurt bijvoorbeeld in de combinaties ff, fi, ffi en fl. Je ziet dat het boogje van de f doorloopt in de letter erna. Bij het ene lettertype is dit duidelijker te zien dan bij een ander. Wanneer ik bijvoorbeeld ffi in een schreefloos lettertype schrijf, ffi, dan zie je dat het dwarsstreepje van de tweede f niet doorloopt. Er wordt dus wel een ligatuur gebruikt alleen lopen de boogjes nu niet in elkaar over.

Het gebruik van ligaturen gaat vanzelf en zit ingebakken in het font dat je gebruikt. Wanneer je geen ligatuur wilt gebruiken op een bepaalde plaats, bijvoorbeeld wanneer je een samenstelling heb, waarbij de ff twee delen aan elkaar plakt, bijvoorbeeld inschrijfformulier. In zo'n geval is het mooier daar geen ligatuur te gebruiken. Je kunt dit voorkomen door \- in te voegen tussen de f'en: `inschrijf\-formulier`. Dit commando geeft eigenlijk een speciale afbreekplaats aan. In (vaak samengestelde) woorden die niet in de standaard afbrekroutines voorkomen, kun je op deze manier plaatsten aangeven waar je het LaTeX mogelijk maakt om het woord af te breken.

Een opvallend detail is dat de ligaturen niet bestaan voor de combinaties met de j. Nu zijn er niet zo heel veel woorden met de combinaties fj, maar toch nog wel heel wat. Of het een schoonheidsfoutje is of iets waar men niet aan gedacht heeft, is niet duidelijk.

Hoofdstuk 3

Documentsindeling

In dit deel leer je alles over het indelen van een document. Daarmee bedoel ik de manier waarop je hoofdstukken, paragrafen, sectie, e.d. kunt aanbrengen, maar ook de manier waaarop je grote stukken van je document onder kan verdelen.

3.1 Sectioning

Het belangrijkste onderdeel van een document is zonder twijfel de inhoud. Maar op een goede tweede plaats komt toch wel de manier waarop de inhoud is onderverdeeld in behapbare delen. In een boek gebruik je meestal hoofdstukken, in een artikel secties. Dit onderverdelen noem je sectioning.

In LaTeX heb je natuurlijk ook de beschikking over dit soort functies. Al hangt de exacte invulling van de commando's af van de documentclass die je gebruikt. De drie classes die je het meest gebruikt zijn 'article', 'report' en 'book'.

3.1.1 article

De documentclass 'article' is bedoeld voor kleine artikelen en verslagen. In de sectioningmogelijkheden is daar ook rekening mee gehouden. Een gedeelte van je artikel noem je een sectie en die geeft je zo aan:

```
\section{titel}
```

Dit commando maakt een kopje voor je aan waarin de titel van de sectie wordt weergegeven. Het kopje 'Sectioning' op deze pagina is een voorbeeld van een section.

De witruimte die valt tussen het einde van de tekst en het kopje is in principe variabel en wordt door LaTeX gebruikt om ervoor te zorgen dat voetnoten, figuren, formules niet maar half op de pagina gezet worden. Je kunt dit nauwelijks beïnvloeden, maar over het algemeen gaat LaTeX goed om met dit soort zaken.

Het commando `\section{}` geeft je sectie ook meteen een nummer, opdat het opgenomen kan worden in een inhoudsopgave. Het is mogelijk om een sectie niet te nummeren, dat doe je door een asterisk toe te voegen:

```
\section*{titel}
```

De sectie wordt dan ook niet in de inhoudsopgave opgenomen. Het is niet één, twee, drie mogelijk om de nummers van de secties weg te laten en de sectie toch in de inhoudsopgave te vermelden. Om dit te doen zul je het commando dat de secties aanmaakt (`\section{}`) en dat de inhoudsopgave aanmaakt (`\tableofcontents`) moeten herschrijven. Dit houdt in dat je een nieuwe documentclass moet schrijven en daar moet je Sven voor heten.

Naast de secties kent de documentclass 'article' nog twee subniveau's:

```
\subsection{titel}
```

en

```
\subsubsection{titel}
```

Deze commando's bestaan natuurlijk ook altijd in een gesterde[1] versie, waardoor de sub(sub)sectie niet wordt genummerd en ook niet in de inhoudsopgave wordt vermeld. Belangrijk is wel om te weten dat wanneer je een gesterde versie gebruikt, dat de desbetreffende (sub(sub))sectie wordt overgeslagen in de nummering van secties. Wanneer je dus twee secties hebt, waarvan de tweede gesterd, dan zullen alle subsecties van de tweede sectie worden genummerd alsof ze bij de eerste horen. Wanneer je een hele sectie niet wilt nummeren dan zul je ook alle subsecties en subsubsecties moeten sterren.

Het is mogelijk een niveau over te slaan, bijvoorbeeld door na een sectie een subsubsectie te beginnen. Dit zie je natuurlijk wel terug in de nummering. Wanneer je namelijk een subsubsectie invoert direct na de eerste sectie dan zal deze het nummer 1.0.1 krijgen; dat is ook wel logisch want voor de eerste subsectie zit natuurlijk de nulde subsectie!

Als het goed is heb je zelden of nooit zoiets nodig als een subsubsubsection. Wanneer je dat wel nodig denkt te hebben dan moet je je om te beginnen even achter de oren krabben en er ernstig over nadenken of het niet beter is om je verhaal anders op te bouwen. Een mogelijkheid is dat je project eigenlijk zo groot is dat het predikaat artikel eigenlijk niet meer van toepassing is. Je kunt dan overgaan op een andere documentclass.

Wanneer je onverhoopt zo eigenwijs bent dat je toch een subsubsubsection nodig denkt te hebben[2], dan moet je zelf zo'n environment schrijven. Het schrijven van environments komt later nog aan de orde.

3.1.2 report

De documentclass 'report' is in principe bedoeld voor het maken van grote verslagen, die meerdere hoofdstukken kunnen bevatten. Voor deze cursus gebruik ik bijvoorbeeld de class 'report'.

Een van de belangrijkste facetten van deze class is de uitbreiding met chapters. Een chapter is een hoofdstuk en het valt in principe boven de section. Een chapter kan dus meerdere sections bevatten. Het commando is

[1] of is het geasteriskte...

[2] In eerste instantie ben ik begonnen om de afzonderlijke hoofdstukken van deze cursus te schrijven met de class 'article'. Precies omdat ik dan in dit hoofdstuk het kopje 'article' als subsubsubsection moest beschrijven, heb ik besloten de class 'report' maar te gebruiken.

`\chapter{`*titel*`}`

Wanneer je de opmaak van de kop van dit hoofdstuk, op pagina 17, bekijkt dan zie je dat het commando `\chapter{}` meer doet dan regel met een klein kopje met nummering maken. Zoals je ziet krijg je een kop 'Hoofdstuk 3', een witruimte en dan de titel van het hoofdstuk. LaTeX zorgt er tevens automatisch voor dat een hoofdstuk altijd op een nieuwe pagina begint.

Er bestaat ook een gesterde versie van chapter:

`\chapter*{`*titel*`}`

Deze laat de kop 'Hoofdstuk #' weg en vermeld alleen de naam van het hoofdstuk. Het is echter zeer de vraag of je die ooit nodig zult hebben, omdat je toch op z'n minst de hoofdstukken wil vermelden in je inhoudsopgave. Realiseer je ook dat wanneer je een chapter stert, dat je ook alle onderliggende secties, subsecties, e.d. moet sterren. In de class 'report', en in 'book' trouwens ook, krijgen de subsubsecties automatisch geen nummer meer. Deze kleinste kopjes zijn ook nauwelijks de moeite waard om op te nemen in je inhoudsopgave. Gedrochten als subsubsectie 1.1.1.1 zijn natuurlijk niet om aan te zien.

3.1.3 book

Wanneer je een nog groter project onder handen hebt kun je gebruik maken van de documentclass 'book'. Deze heeft boven chapter nog een onderverdeling:

`\part{`*titel*`}`

Zoals de naam al aangeeft is `\part{}` een gedeelte van een boek; het levert een nieuwe rechterpagina met daarop het nummer en de naam het deel. Tevens zorgt de class 'book' ervoor dat nieuwe hoofdstukken automatisch op een nieuwe rechterpagina worden opgestart. Dit houdt in dat na het schutvel van het gedeelte een lege linkerpagina volgt met daarna op de rechterpagina het eerste hoofdstuk dat valt onder dat deel.

De indeling van een niveau verandert (vrijwel) niets wanneer je in een andere class een hoger niveau gaat gebruiken. Hiermee bedoel ik dat wanneer je verschillende artikelen (met de class 'article' gemaakt) wilt bundelen in een report, dan volsta je door de body[3] van de afzonderlijke documenten samen in een TeX-bestand te zetten en van de afzonderlijke titels, die je hebt opgegeven in de afzonderlijke preambles, de titels van chapters te maken.

3.2 Het gebruik van inputfiles

Je begrijpt dat wanneer de afzonderlijke artikelen al omvangrijk waren dat de TeX-file van de samengevoegde artikelen enorm groot gaat worden. Dit kan zelfs zo de pan uit rijzen dat je editor er enorm traag van wordt. Om dit soort dingen op te vangen kun je in LaTeX gebruik maken van inputfiles.

Dit houdt in dat je je brontekst in stukken hakt en de delen onderbrengt in apparte bestandjes. Met behulp van het commando

[3] Dat deel tussen `\begin{document}` en `\end{document}`

```
\input bestand.tex
```

laat je de compiler weten dat hij op dit punt verder moet gaan met de eerste regel van het bestand 'bestand.tex' (de toevoeging .tex is niet noodzakelijk). De compiler plakt vervolgens keurig netjes de delen aan elkaar. Om dit te illustreren heb ik de bestanden van deze cursus zo aangepast dat we nu een bestand 'cursus.tex' hebben en daarnaast van elke afzonderlijke les ook een bestand. Het enige wat je hoeft te doen is de delen met behulp van `\input` aan elkaar plakken en cursus.tex door de compiler te halen. In de preamble van 'cursus.tex' dienen natuurlijk wel alle packages voor *alle* inputfiles geladen moeten worden; ook nieuwe commando's en environments moeten voor *alle* inputfiles vermeld worden.

Wanneer je met een groot bestand bezig bent en je wil snel controleren of datgene waar je mee bezig bent er goed uitziet, dan moet je wachten totdat alles opnieuw gecompiled is. Hoe groter je bestand wordt hoe langer dit duurt. Wanneer je gebruik maakt van inputfiles kun je dit heel makkelijk oplossen. Je zet alle niet ter zake doende inputfiles gewoon achter een %, je maakt er eigenlijk een commentaarregel van. Bij het schrijven van dit deel van de cursus, ziet een deel van mijn 'cursus.tex' er zo uit

```
\begin{document}
\maketitle
\input proloog.tex
\tableofcontents
%\input les01.tex
%\input les02.tex
\input les03.tex
\end{document}
```

Het gebruik van inputfiles is ook erg handig wanneer je met meerderen aan een document wil werken. Het is namelijk onmogelijk om tegelijk in een bestand te werken. Een van de twee is dan altijd read-only. Wanneer je de inhoud splitst kun je allebei aan hetzelfde document werken zonder dat je in elkaars vaarwater vaart[4].

[4]Bij de Impuls doen we dit ook en hebben we zelfs voor ieder een persoonlijke testomgeving zodat we ook de resultaten tegelijk kunnen waarnemen

Hoofdstuk 4

Tabellen en figuren

4.1 Floating bodies

Heel vaak wil je in je document een plaatje of een tabel opnemen. Het probleem is dan vaak waar je zo'n object invoegt. Het liefst wil je het object zo dicht mogelijk bij het beschrijvende stuk tekst, maar je wil ook niet dat de ene helft op een pagina komt te staan en de ander op een ander deel. Kortom, er komen een heel groot aantal zaken kijken bij het plaatsen van een plaatje of een tabel.

LaTeX lost dit in een keer voor je op door gebruik te maken van zogenaamde *floating bodies*. Dit zijn objecten die LaTeX invoegt op plaatsen waar er genoeg ruimte is of gemaakt kan worden. Deze interne selectieprocedure zorgt er dan automatisch voor dat er een plaatje niet half op een pagina komt of dat het plaatje er wel opkomt maar het onderschrift pas op de volgende pagina terecht komt. Er zijn twee soorten flaoting body environments: *figure* en *table*, die je voor het geval *table* op de volgende manier begint:

```
\begin{table}[placement specifier]
```

en afsluit met

```
\end{table}
```

Het belangrijke stuk is hierin de *placement specifier*. Dit is een serie parameters waarmee je kunt aangeven wat jij belangrijk vindt m.b.t. de plaatsing van het betreffende object. Er zijn vier verschillende opties:

h wat betekent here,
b wat betekent bottom,
t wat betekent top,
p wat betekent page

Elk van deze letters staat voor een bepaalde plaats waar het object geplaatst kan worden. Van links naar rechts probeert LaTeX aan de specifiers te voldoen. Wanneer het met de eerste niet lukt probeert hij de tweede, etc.

Speciale aandacht vraagt de placement specifier `p`. Wanneer een object op zo'n rare plaats staat (daarmee bedoel ik een plaats waar weinig ruimte is: bijvoorbeeld door kopjes, paginaeindes, voetnoten, andere objecten, etc.) dat LaTeX het eigenlijk niet netjes kwijt kan, dan worden ze opgespaard totdat

er zoveel objecten zijn dat er een apparte 'floating body'-pagina kan worden gemaakt. Dat is dus een pagina die ergens tussen de tekst wordt ingevoegd waar een aantal opgespaarde objecten opstaan. Met het commando `\clearpage` kun je LaTeX de opdracht geven om een pagina leeg te maken en alle objecten in de wachtrij te plaatsen en een nieuwe pagina te beginnen.

Daarnaast is er de mogelijkheid om een uitroepteken aan de placement specifier toe te voegen. Dit betekent dat LaTeX een klein oogje dicht moet knijpen bij het beoordelen of iets nog wel kan of niet.

Ik zal de werking van de placement specifiers wat verder uitleggen aan de hand van een voorbeeld. Het opgeven van de opties `[!hbp]` betekent zoiets als: probeer als het enigzins mogelijk is het object hier (h – here) te plaatsen. Als dat niet lukt, kijk dan of je het object onder aan de pagina kan plaatsen (b – bottom). Als ook dat niet lukt (omdat er blijkbaar echt geen plaats gemaakt kan worden) spaar de flaoting bodies dan op totdat je een hele pagina (p – page) gevuld hebt en plaats die speciale floating-bodypagina er dan maar tussendoor. En ik vind het helemaal niet zo erg als het er een beetje raar uitziet (!).

De floating bodies zijn eigelijk alleen maar omhulsels. De inhoud moet nog toegevoegd worden. De floating body 'table' gebruik je om hem daarna te vullen met een tabel en 'figure' om hem daarna te vullen met een figuur. Wanneer je deze environments gebruikt is het daarna ook mogelijk om een lijst met tabellen en een lijst met figuren te maken, zoals je ook de inhoudsopgave hebt. Dit doe je met resp. `\listoftables` en `\listoffigures`.

4.2 Tabellen

Het environment dat tabellen beschrijft heet `tabular`. Dat is dus iets anders dan `table`; immers de floating body table wordt gevuld met een tabular. Belangrijk bij een tabel is hoeveel kolommen en rijen je nodig hebt. Bij een tabular geef je het aantal klommen aan in de opties bij het begin van je tabular. Dit doe je door op te geven hoe een bepaalde kolom uitgelijnd moet worden: l(eft), r(ight) of c(enter). Daarbij kun je tussen elke kolom aangeven wat het kolomscheidingsteken is, bijvoorbeeld:

```
\begin{tabular}{l | r @{scheidingswoord} c |}
```

Wanneer je als scheidingsteken '`|`' gebruikt worden de verticale lijnen automatisch doorgetrokken. Een horizontale lijn tussen twee rijen voeg je toe met `\hline`. Voor het gebruik van dubbele lijnen met doorgetrokken verticalen e.d. is er het package 'hhline', documentatie hierover is op internet te vinden..

De rijen voeg je gewoon regel voor regel toe. Je begint met de tekst voor de eerste rij en je geeft aan dat de volgende kolom begint door een & in te voegen[1]. Een rij sluit je af met \\, het normale symbool om een regel af te breken. Let er op dat je evenveel & hebt als scheidingstekens, ook al is de rest van de rij leeg. De laatste rij van de tabular hoef je niet af te sluiten met \\, maar natuurlijk wel als er daarna alleen nog maar een horizontale lijn komt.

Ik kan me voorstellen dat dit niet geheel duidelijk is. Daarom hieronder een voorbeeld dat je kunt zien in tabel 4.1:

[1]dat is de reden dat je het teken & niet zomaar als & op kan nemen als broncode. Dit teken heeft namelijk de functie van kolomscheiding is het tabular environment. Je gebruikt daarvoor in de plaats `\&`.

```
\begin{table}[!hb]
\caption{Een voorbeeld tabel}
\label{voorbeeldtabel}
 \begin{center}
  \begin{tabular}{| l | r | c @{ * } l |}
  \hline
  links & rechts & centraal    & mooi    \\
  een   & tabel  & met         & leuke   \\
  rijen & zonder & horizontale & lijnen  \\
  \hline
  of    & met    & horizontale & lijnen  \\
  \hline
  wat   & je     & maar        & wil     \\
  \hline
  zelfs & \multicolumn{2}{c}{samengevoegde} & rijen \\
  \hline
  erg   &        & leuk        & allemaal \\
  \hline
  \end{tabular}
 \end{center}
\end{table}
```

Tabel 4.1: Een voorbeeld tabel

links	rechts	centraal	* mooi
een	tabel	met	* leuke
rijen	zonder	horizontale	* lijnen
of	met	horizontale	* lijnen
wat	je	maar	* wil
zelfs	samengevoegde		rijen
erg		leuk	* allemaal

Ik heb behalve de invulling van de tabel een paar andere dingen in dit voorbeeld opgenomen.

Ten eerste het gebruik van labels en referenties. LaTeX nummert alles wat er te nummeren valt automatisch voor je: paginanummers, vergelijkingen, figuren, tabellen, noem maar op. Wanneer je ergens een vergelijking tussenvoegt wordt de nummering automatisch aangepast. Dit heeft tot gevolg dat het niet voldoende is om naar een nummer te referen, immers dat zou nog kunnen veranderen. Je moet daarom datgene waar je naar wil verwijzen eenmalig een label geven. Dat doe je met `\label{`*naam*`}`. Vervolgens kun je met `\ref{`*naam*`}` verwijzen naar het overeenkomstige label. Wanneer ik bijvoorbeeld naar de voorbeeldtabel wil verwijzen doe ik dat met `\ref{voorbeeldtabel}` en er komt te staat 4.1.

LET OP: om de juiste nummers bij de juiste referenties te krijgen moet LaTeX opslaan bij welke naam welk nummer heeft, opdat de nummers de volgende keer worden ingevuld. Je moet dus wanneer je labels of referenties veranderd hebt, je

file altijd twee keer compilen. Wanneer het nodig is een tweede keer te latexen, zal LaTeX hiervoor waarschuwen.

Het is overigens een goed idee om, wanneer je tabellen of figuren labelt, een label te geven aan de titel (engels: caption, vandaar het commando `\caption{titel}`). Je kunt het ook echt aan de tabel hangen, maar dat gaat soms wel goed en soms niet. Dit werkt eigenlijk altijd.

Het tweede nieuwtje betreft het gecentreerd uitlijnen van text of in dit geval een tabel. Hier bestaat het environment `center` voor. Zoals je ziet werkt dat ook voor gewone tekst.

Voor alleen links uitgelijnde tekst bestaat het environment `flushleft`

en voor rechts uitgelijnde tekst `flushright`.

Als laatste wil ik de aandacht vestigen op het feit dat je n kolommen kan samenvoegen met het commando `\multicolumn{n}{`*`justify`*`}{}`. Uit het voorbeeld is denk ik wel duidelijk hoe dat werkt. Kolommen splitsen kan naar mijn weten niet.

4.3 Figuren

Soms wil je wel eens een plaatje in je document opnemen. Dat kan van alles zijn: een foto, een tekening of een grafiekje dat je bijvoorbeeld uit gnu-plot haalt. Deze plaats je dan in het environment `figure`. Op de systemen die je op de uni vindt is het eigenlijk alleen van belang (het geeft ook het mooiste resultaat) om eps'jes in te voegen.

Een eps'je is een encapsulated postscript en het is een bepaald bestandsformaat voor plaatjes, zoals jpg, gif, tif, fig, en bmp dat ook zijn. Met programma's als correldraw, xv, xfig en gimp (de laatste drie zijn op de uni geïnstaleerd) kun je plaatjes in eps-formaat exporteren.

Om je eps'je vervolgens te kunnen importeren in je tex-bestand moet je de package *graphicx* laden en met behulp van het commando

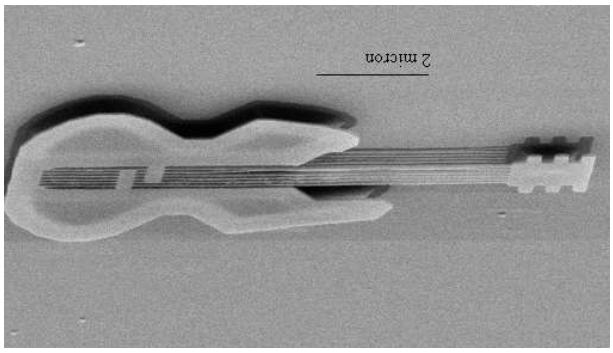
`\includegraphics[`*`key=value,...`*`]{`*`file`*`}`

kun je het plaatje invoegen. De keys die je als opties mee kan geven zijn erg handig om je plaatje op de gewenste plaats te krijgen. Zo kun je met `width` en `height` het formaat bepalen (wanneer je een van de twee opgeeft wordt de andere mee veranderd zodat de verhouding correct blijft, als je ze alle twee opgeeft worden ze natuurlijk beide aangepast). Met `scale` kun je de vergrotingsfactor aangeven (tussen 0 en 1, natuurlijk) en `angle` geeft de rotatiehoek. De opties worden van links naar rechts uitgevoerd.

In het voorbeeld van figuur 4.1 is dit gedemonstreerd voor het overigens bekend veronderstelde plaatje *nanotar.eps*:

```
\begin{figure}[!hb]
 \begin{center}
 \includegraphics[angle=180, width=8cm]{nanotar.eps}
 \caption{H\'e, dit plaatje staat op zijn kop!}
```

```
 \label{voorbeeldfiguur}
 \end{center}
\end{figure}
```



Figuur 4.1: Hé, dit plaatje staat op zijn kop!

Overigens wordt xdvi erg langzaam van die plaatjes (al gaat het op de nieuwe sunblades in TK8 een stuk sneller). Je kunt met het xdvi-window actief op de 'v'-toets drukken om het scannen van plaatjes te skippen. Soms krijg je dan een raar boxje te zien, dat duidelijk de afmeting van het plaatje weer zou moeten geven maar dat absoluut niet doet. Dit heeft iets te maken met boundingboxen van postscriptbestanden en daar heeft alleen Wilbert verstand van.

Hoofdstuk 5

Wiskunde

5.1 Mathmode

Voor het opmaken van wiskundige uitdrukkingen gelden vanzelfsprekend hele andere criteria als voor gewone tekst. In LaTeX moet je daarom van de gewone, standaard mode naar de speciale *mathmode* gaan. Dit kan op een aantal manieren, afhankelijk van wat je wilt.

De eerste manier is door je uitdrukking tussen $'s te zetten. LaTeX springt dan meteen in de mathmode en behandelt alles wat tussen de $'s staat als wiskundige uitdrukking. Het gebruik van deze methode is uitermate geschikt als je uitdrukking maar kort is en bovendien midden in de normale tekst staat. Bijvoorbeeld wanneer je aan het uitleggen bent wat bepaalde symbolen in een bepaalde formule betekenen of wanneer je een formule zomaar tussendoor geeft, zoals in:

> 'Het is natuurlijk bij iedereen allang duidelijk dat de stelling van Pythagoras ($a^2 = b^2 + c^2$) een speciaal geval is van de cosinusregel ($a^2 = b^2 + c^2 - 2bc\cos\alpha$).'

De tweede manier is door bepaalde math-environments te gebruiken. Bijvoorbeeld het `displaymath` environment, dat alles wat er staat tussen de bijbehorende `\begin{}` en `\end{}` statements als wiskundige uitdrukkingen beschouwt. Andere voorbeelden zijn de environments `equation` en `eqnarray`.

5.2 Letters en symbolen

5.2.1 Grootheden, eenheden en constanten

In de natuurkunde maak je gebruik van symbolen om te verwijzen naar zaken die een bepaalde betekenis hebben. Zo kan de letter A staan voor een punt in een vlak, maar ook voor de eenheid Ampère of voor de elektromagnetische vectorpotentiaal. Vaak blijkt uit de context wel wat een specifieke letter betekent (in een goede tekst wordt het ook altijd vermeld), maar er zijn toch een aantal richtlijnen aan te geven over hoe bepaalde symbolen worden afgedrukt.

Grootheden worden meestal schuingedrukt; aangezien in formules bijna alleen maar grootheden staan, worden in mathmode daarom standaard alle tekens

cursief gedrukt. Eenheden daarentegen staan normaal. Wanneer je bijvoorbeeld wilt schrijven dat een object een massa heeft van één kilo, ziet dat er dus zo uit: $m = 1\,\mathrm{kg}$. De m is de grootheid en staat dus cursief, kg is de eenheid en staat dus normaal. Om in mathmode een bepaald deel normaal (dus *roman*) te schrijven gebruikt je het commando `\mathrm{}`. Constanten in formules staan net zoals eenheden gewoon rechtop, immers ze dienen slechts om beide zijden van een vergelijking dezelfde eenheid te geven.

5.2.2 Indices

Je maakt een subindex door een `_` te gebruiken: `$F_\mathrm{mpz}$`, wat F_mpz levert, is hier een voorbeeld van. Wanneer je meerdere symbolen als subscript wil gebruiken zet je het tussen accolade's achter zo'n `_`. Bijvoorbeeld `$F_\mathrm{middelpuntzoekend}$` wat $F_\mathrm{middelpuntzoekend}$ oplevert. Een superindex maak je op overeenkomende wijze, alleen nu gebruik je een `^`. Er gelden verder dezelfde regels als voor de subindices.

Afhankelijk van waar een index naar verwijst schrijf je hem rechtop of cursief. Wanneer hij verwijst naar een eenheid of zomaar wat informatie geeft is hij rechtop. Je schrijft dus F_mpz in plaats van F_{mpz}. Wanneer echter een index verwijst naar een grootheid, komt hij natuurlijk wel cursief te staan: de kracht in de z-richting schrijf je als F_z, maar de zwaartekracht als F_z. Dit onderscheid lijkt misschien wat overdreven, maar het maakt uitdrukkingen als $F_{\mathrm{z}z}$ wel duidelijk: de zwaartekracht in de z-richting!

5.2.3 Vectoren

Vectoren kunnen op meerdere manieren worden aangegeven. Sommige auteurs kiezen ervoor om er pijltjes boven te zetten ($\vec{A}$ door `\vec{A}`), andere kiezen ervoor om ze vet te drukken ($\mathbf{A}$ door `\mathbf{A}`). Weer andere zetten er een streepje boven of onder, maar de meesten vinden dat stom.

5.2.4 Wiskundige operatoren en andere symbolen

Je moet wel een heel obscuur soort wiskunde bedrijven, dat je symbolen tegenkomt die in LaTeX niet bestaan. Een heleboel staan vermeld in de *The Not So Short Introduction To LaTeX2e*, en anders kun je op internet zoeken naar het gewenste font. Vaak zijn de namen van de commando's zo gekozen dat je ook begrijpt wat het betekent. Belangrijk zijn dingen als sommaties, integralen en breuken. Je vormt deze symbolen door de commando's `\sum`, `\int` en `\fraq{}{}` respectievelijk. De implementatie is verder logisch al lijkt het in je broncode vaak een warboel. Zie verder de voorbeelden verderop.

5.2.5 Haakjes

Er zijn een heleboel verschillende soorten 'haakjes', of tekens die een vergelijkbare functie hebben. Een aantal vind je gewoon op je toetsenbord, zoals ronde haakjes, accolades[1] en rechte haakjes. Voor de meesten moet je echter een bepaald commando gebruiken, zie ook *The Not So Short Introduction To LaTeX2e*.

[1] let erop dat je deze al een betekenis hebben in LaTeX, zodat je ze van een backslash moet voorzien: \{ en \}

Haakjes heb je ook in verschillende grootten. Naar mate er meerdere haakjes in elkaar haken, worden de haakjes steeds groter. In LaTeX gebeurt dat automatisch door elk haakje te vergezellen door een `\left` of `\right`. Soms kan het voorkomen dat je het niet eens bent met de grootte die LaTeX voor je kiest, of omdat je een slechte smaak hebt of omdat je gewoon eigenwijs bent. Dan kun je je haakje vergroten door deze commando's, van klein naar groot: `\big`, `\Big`, `\bigg` en `\Bigg`. Zie verder voorbeelden.

5.2.6 Matrices

Een matrix is niets anders dan een tabelletje met getallen omgeven door haakjes. Dat is zo in de wiskunde en ook in LaTeX. Om een matrix te maken gebruik je daarom het environment `array` dat net zoals `tabular` gevolgd wordt door een tekenreeks die aangeeft hoe de elementen moeten worden uitgelijnd. De elementen worden gescheiden door een `&` en een regel sluit je af met `\\`.

De haakjes maak je vervolgens door om het gehele `array`-environment haakjes te zetten met `\left(`, `\right)` of `\left[`, `\right]`, afhankelijk van welke notatie je prefereert. Een voorbeeldje:

```
\begin{displaymath}
\mathbf{A} = \left(
  \begin{array}{c c c}
   a_{11} & a_{12} & \cdots \\
   a_{21} & a_{22} & \cdots \\
   \vdots & \vdots & \ddots \\
  \end{array}
  \right)
\end{displaymath}
```

levert

$$\mathbf{A} = \left(\begin{array}{c c c} a_{11} & a_{12} & \cdots \\ a_{21} & a_{22} & \cdots \\ \vdots & \vdots & \ddots \end{array} \right)$$

5.3 Voorbeelden

Het gaat te ver om alle wiskundige mogelijkheden helemaal te beschrijven. In plaats daarvan geef ik enkele voorbeelden van uitdrukkingen en beschrijf hoe je ze opbouwt. Dan leer je hoe je zelf *The Not So Short Introduction To LaTeX2e* kan gebruiken om dingen uit te vogelen.

5.3.1 Sommaties en breuken

We gaan uit van de uitdrukking voor de correlatie tussen twee variabelen

$$\mathrm{corr}(X,Y) = \frac{\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})}{\left[\sum_{i=1}^{N}(x_i-\overline{x})^2\sum_{i=1}^{N}(y_i-\overline{y})^2\right]^{\frac{1}{2}}} \tag{5.1}$$

Om deze formule te maken gaan we om te beginnen in mathmode. Aangezien dit een vergelijking is gebruiken we hiervoor het environment `equation`.

```
\begin{equation}

\end{equation}
```

Hiertussen komt onze formule te staan. De volgende stap is de 'correlatiefunctie'. Dit is niet het product van variabelen c, o, r en r, zodat we moeten overgaan op het mathrm-font:

```
\begin{equation}
\mathrm{corr}(X,Y) =
\end{equation}
```

Voor veel wiskundige functie bestaat overigens een appart commando, dat naast het juiste fonttype ook meteen de spatiering rond de functie regelt. Voorbeelden zijn: `\sin, \cos, \sec` (zie verder de niet heel korte handleiding).

Vervolgens zien we dat er na het =-teken een breuk volgt. We voegen dus de functie `\frac{}{}` in:

```
\begin{equation}
\mathrm{corr}(X,Y) = \frac{}
                          {}
\end{equation}
```

Waarbij ik voor de duidelijkheid de haakjes die de noemer aangeven vast op een volgende regel heb gezet. We beginnen met het invullen van de teller. Deze bestaat uit de sommatie over iets. We vullen dus een functie `\sum` toe en geven het bereik aan met behulp van sub- en superindices:

```
\begin{equation}
\mathrm{corr}(X,Y) = \frac{\sum_{i=1}^N}
                          {}
\end{equation}
```

Dit ziet er nu zo uit:

$$\mathrm{corr}(X,Y) = \frac{\sum_{i=1}^{N}}{}$$

Zoals je misschien opvalt staan de indices die het bereik aangeven naast de som en niet eronder en erboven. Dit ligt aan de *style* waarin je werkt en je past hem aan door `\displaystyle` toe te voegen voor de uitdrukking waar je de stijl van wilt veranderen. Wanneer je sommatie bijvoorbeeld midden in tekst zou staan is het wel handig als de grenzen naast de sommatie staan omdat LaTeX anders de regelgrootte aan zou passen wat er lelijk is. Het gemiddeldestreepje-streepje voeg je toe door `\overline` zodat de gehele teller er zou uitziet:

```
\begin{equation}
\mathrm{corr}(X,Y) = \frac{\displaystyle
          \sum_{i=1}^N (x_i-\overline x)(y_i - \overline y)}
          {}
\end{equation}
```

Voor de noemer voeren we ook weer `\displaystyle` in en een linker rechte haak `\left[`. Vervolgens de sommatie met de goede grenzen en de uitdrukking tussen haakjes. Voor het kwadraat voer je zoals bekend gewoonweg `^2` toe. Vervolgens op dezelfde manier de volgende sommatie en ook de tweede term tussen haakjes inclusief kwadraat. Je sluit af met een `\right]`. De macht van een half voer je in door `\frac{1}{2}`. Het resultaat komt er nu zo uit te zien:

```
\begin{equation}
\mathrm{corr}(X,Y)=
  \frac{\displaystyle \sum_{i=1}^N(x_i-\overline x)
                                  (y_i-\overline y)}
       {\displaystyle \left[\sum_{i=1}^N(x_i-\overline x)^2
                            \sum_{i=1}^N(y_i-\overline y)^2
                      \right]^\frac{1}{2}}
\end{equation}
```

Dit ziet er zo uit:

$$\mathrm{corr}(X,Y) = \frac{\displaystyle\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})}{\displaystyle\left[\sum_{i=1}^{N}(x_i-\overline{x})^2\sum_{i=1}^{N}(y_i-\overline{y})^2\right]^{\frac{1}{2}}}$$

En dat is precies wat we wilden.

5.3.2 Reeksen

Uitgangspunt is de reeksvergelijking voor de complexe e-macht:

$$\begin{aligned}\exp(ix) &= \sum_{k=0}^{\infty} i^k \frac{x^k}{k!} \\ &= 1 + ix - \frac{x^2}{2!} - i\frac{x^3}{3!} + \frac{x^4}{4!} + i\frac{x^5}{5!} + \cdots \\ &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + i\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots\right) \\ &= \cos(x) + i\sin(x) \end{aligned} \tag{5.2}$$

Wat meteen opvalt is dat deze vergelijking meerdere regels beslaat. Om dit netjes te doen gebruiken we het speciale environment `align`. Dit environment doet ongeveer hetzelfde als het environment `eqnarray` zoals beschreven in *The Not So Short Introduction To LaTeX2e*, echter hierin is de spatiëring beter. Voor het environment `align` moet je het package `amsmath` laden.

Het environment zorgt voor een standaard $\{rl\}$ tabular in mathmode, waarin de displaystyle uit het vorige voorbeeld ook meteen goed is. De eerste kolom gebruiken we voor de uitdrukking $\exp(ix)$ en de tweede voor de werkelijke formules. De kolommen worden gescheiden door `&`-tekens en een regel sluit je af met `\\`. De eerste regel beslaat verder alleen een sommatie en de regels daarvoor zijn in het vorige voorbeeld besproken. Het is gewoon een kwestie van de andere regels toevoegen. Bij de tweede regel dien je de eerste kolom leeg te laten en in de tweede staat een =-teken gevolgd door de uitdrukking: `1 + ix - \frac{x^2}{2!} - i\frac{x^3}{3!}`, etc. Denk eraan dat je in je broncode je meerdere regels kunt gebruiken voor deze reeks, terwijl LaTeX pas op een andere regel verder gaat na de `\\`. Ook zul je zien dat LaTeX elke regel nummert als een apparte formule, wanneer je dat niet wil, zoals in dit voorbeeed dan zet je voor de `\\` een `\nonumber`.

Op overeenkomstige wijze voeg je alle regels toe en je krijgt het volgende eindresultaat:

```
\begin{align}
 \exp(ix) & = \sum_{k=0}^\infty i^k\frac{x^k}{k!} \nonumber \\
          & = 1 + ix - \frac{x^2}{2!} - i\frac{x^3}{3!}
                + \frac{x^4}{4!} + i\frac{x^5}{5!}
                + \cdots \nonumber \\
           & = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots
                + i\left(x - \frac{x^3}{3!} + \frac{x^5}{5!}
                + \cdots\right) \nonumber \\
           & = \cos(x) + i\sin(x)
\end{align}
```

En dat ziet er zo uit:

$$\begin{aligned}
\exp(ix) &= \sum_{k=0}^{\infty} i^k \frac{x^k}{k!} \\
&= 1 + ix - \frac{x^2}{2!} - i\frac{x^3}{3!} + \frac{x^4}{4!} + i\frac{x^5}{5!} + \cdots \\
&= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + i\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots\right) \\
&= \cos(x) + i\sin(x)
\end{aligned} \tag{5.3}$$

Nogmaals: in je broncode kun je, zoals hier, meerdere regels gebruiken voor een regel terwijl in je document er maar een regel gebruikt wordt.

Hoofdstuk 6

Extra's

Wanneer je de vorige vijf hoofdstukken hebt doorgewerkt dan weet je al voldoende om een practicumverslag tot in de puntjes te verzorgen. Wanneer je LaTeX echter vaker en vaker gaat gebruiken, zijn er bepaalde extra's die je misschien ooit wil toepassen. Daar wil ik het in dit hoofdstuk over hebben.

6.1 Nieuwe commando's

Binnen een tekst die over een bepaald onderwerp gaat, komen vaak veelvuldig dezelfde uitdrukkingen voor. Je kunt denken aan een boekbespreking waar de titel van het boek vaak genoemd wordt, of aan een wiskundescriptie waar bepaalde wiskundige uitdrukkingen vaak worden gebruikt. Maar ook speciale functies die je specifiek voor het gebruik in een bepaald document ontwerpt, vallen hieronder. Het is dan mogelijk om deze standaardzaken te verpakken in zelfgedefinieerde, nieuwe commando's. Dit doe je in de preamble van je document met:

```
\newcommand{naam}[n]{inhoud}
```

Op deze manier creëer je het commando `\naam` dat inhoud doet en n argumenten heeft. Een goed voorbeeld is het vereenvoudigen van ingewikkelde of onoverzichtelijke wiskundige uitdrukkingen.

Neem bijvoorbeeld een tekst over quantummechanica. Daarin werk je meestal met de Dirac-bracketnotatie om toestanden aan te geven. Dit ziet er uit als $|\psi\rangle$ voor de bra van toestand ψ en $\langle\psi|$ de bijbehorende cket. In broncode ziet dat eruit als `$|\psi\rangle$` en `$\langle\psi|$`. Ik kan me voorstellen dat je dit wil vereenvoudigen. Een manier om dit te doen is door de volgende newcommands toe te voegen:

```
\newcommand{bra}[1]{$|#1\rangle$}

\newcommand{cket}[1]{$\langle#1|$}
```

zodat je in je tekst alleen maar `\bra{\psi}` en `\cket{\psi}` hoeft te typen.

Zoals je ziet kun je het argument $i \leq n$ gebruiken door `#i`. Wanneer een commando al bestaat, dan kun je met `\renewcommand` het commando herdefiniëren; je gebruikt het op dezelfde manier als `\newcommand`. Je moet hier natuurlijk

wel mee opletten! Je moet zeker weten dat als je een commando herdefinieert, je geen vitaal commando van LaTeX hernoemt, want dan werkt het niet meer. Als er op die manier wat mis gaat is het voldoende om een andere naam voor je eigen commando te kiezen.

Naast nieuwe commando's kun je natuurlijk ook nieuwe environments maken. Dat doe je door:

```
\newenvironment{naam}[n]{begin}{end}
```

Nu kun je dus met `\begin{naam}` het environment aanroepen. Wanneer je het environment aangeroepen hebt worden eerst de statements van het `begin`-deel uitgevoerd. Bij het end-statement van het environment worden de statements van het `end`-deel uitgevoerd. Je kunt dit bijvoorbeeld gebruiken om veel voorkomende genestelde environments onder één noemer te brengen.

De argumenten werken op dezelfde manier en de inhoud van de parameters komen tussen accolades direct na de sluitaccolade van het `begin{naam}`-statement, net zoals de placementspecifiers van de floating-bodies. Wanneer een environment al bestaat kun je het herdefiniëren met `\renewenvironment`.

6.2 Je eigen packages

Zoals ik al eerder heb toegelicht heb je voor speciale dingen speciale packages nodig, waar de functies geladen worden die je nodig hebt. Ook heb je net gezien hoe je zelf commando's kan maken. Het ligt dus voor de hand om deze dingen te combineren en te kijken hoe je zelf een package met je eigen commando's kunt maken. Maar voor ik dat uitleg, even nog iets over de bedoeling en het nut van packages.

Een package is handig omdat je bepaalde commando's die je zelf hebt gedefinieerd in meerdere documenten wil gebruiken. Het is dan makkelijk om de definities van die commando's in een package te hebben staan, zodat je alleen maar het package hoeft te laden in het relevante document. Bijkomend voordeel is dat anderen er ook gebruik van kunnen maken (zie verder beneden).

Maar het is niet altijd nodig om een package te maken. Voor deze cursus bijvoorbeeld heb ik heel wat extra dingen moeten maken om LaTeX-commando's te kunnen laten zien zonder dat LaTeX ze als commando's interpreteert. Maar die aanpassingen heb ik hierna nooit meer nodig en dus kan ik ze best (ook al is het een hele lijst) in de preamble laten staan. Natuurlijk heb ik, als voorbeeld, wel een package gemaakt, maar dat is eigenlijk zonde van de moeite geweest.

6.2.1 Een package maken

Een package is een bestand met de extensie .sty. Je maakt dus om te beginnen een bestand 'packagenaam.sty'. In de eerste regel van dit packagebestand moet je aangeven bij welke (kortere naam) het package moet worden aangeroepen. Dit doe je door (let op de hoofdletters!):

```
\ProvidesPackage{naam}
```

Na deze regel kun je commando's gaan invoegen die je in je package wil hebben. Deze commando's moeten er net zo uitzien als in de preamble van een document (wat heet, het package, en dus de definities er in, worden in de preamble

van een document gelezen dus eigenlijk *staan* ze gewoon in de preamble van een document). Om nu het package te laden in een document hoef je alleen maar `\usepackage{naam}` op te nemen in de preamble van het desbetreffende document!

6.2.2 Je package delen met anderen

Wanneer je je prachtige nieuwe commando's gedefinieerd hebt, wil je anderen er vast ook van laten genieten. Het spreekt vanzelf dat je die anderen moet laten weten waar je sty-bestand te vinden is en dus zet je je sty-bestand zo dat de ander er bij kan (door het bestand bijvoorbeeld leesbaar te zetten in een dir op naam van een gedeelde group; een voorbeeld daarvan zijn de sty-bestanden in de dir `/vol/impuls/marcur/tex/`).

Maar hiermee is niet alles gezegd, want hoe weet LaTeX nu waar hij het bestand waar om gevraagd wordt in het usepackage-commando kan vinden? Het antwoord op deze vraag moet niet worden gezocht in de context van LaTeX maar die van het systeem waarop LaTeX draait. In het geval van de systemen op de universiteit maak je gebruik van de LaTeX dat is geïnstalleerd voor gebruik op bijvoorbeeld studs1 of een —blad uit TK8. Je hebt op die apparaten je eigen homedir zoals je weet.

In je homedir bevindt zich nu een bestand '.cshrc'. Dit bestand bevat onder andere de informatie die instelt waar 'gezocht' gaat worden naar bepaalde bestanden. Een klein stukje van een .cshrc ziet er bijvoorbeed zo uit:

```
set path=($path /vol/impuls/bin
                /vol/impuls/marcur/bin
                /usr/local/jpeg/bin)

setenv TEXINPUTS .\:/vol/impuls/lib\:
```

Wat je ziet is dat er paden (directories) worden opgegeven waar naar commando's (in het stuk 'set path') en naar tex-inputbestanden (in het stuk 'setenv TEXINPUTS') wordt gezocht. Merk op dat de eerste plek waar naar tex-inputs wordt gezocht, de huidige dir is (d.w.z. de dir waarvanuit LaTeX wordt aangeroepen). Een package dat in dezelfde dir staat als het bestand wat om het package vraagt, wordt dus altijd gevonden.

Om je zelfgeschreven package te delen met anderen, moet je er dus voor zorgen dat het op een plek staat waar jullie allebei bij kunnen en tevens dat de anderen de dir waar het bestand staat hebben opgenomen in hun 'zoek daar naar tex-inputbestanden'-lijst.

6.3 Het package `fancyhdr`

Een veelgebruikt package is het package `fancyhdr`. Dit package steltje in staat om zelf kop- en voetteksten te definiëren. Om het te gebruiken moet je het natuurlijk includen met het usepackage-commando. Daarnaast moet je LaTeX laten weten dat je ook die apparte paginaopmaak wil gebruiken. Dit doe je door

```
\pagestyle{fancy}
```

Vervolgens moet je gaan aangeven hoe je de header en footers wil zien. Dit doe je door de tekst van de kop- en voetteksten in te stellen met het commando:

```
\fancyhf[specs]{contents}
```

Met behulp van de parameters in *specs* geef je aan welke header of footer je bedoelt. Hiervoor zet je de gewenste combinatie van letters achtereen: H voor header of F voor footer, gevolgd door R voor de rechter, C voor de gecentreerde of L voor de linker kop-/voettekst, gevolgd door O voor de kop-/voettekst op de onevenpagina's of E voor die op de even pagina's (relevant voor tweezijdige documenten).

Het weglaten van een van de drie categoriën opties zorgt ervoor dat het voor alle gevallen van de desbetreffende categorie geldt. Het leegmaken van alle header en footers doe je dus met:

```
\fancyhf{}
```

immers dan geldt voor *alle* gevallen dat de tekst leeg is.

Een ander voorbeeld is het instellen van een paginanummer (opgeslagen in de variabele `\thepage`) bovenin de twee buitenste hoeken van de pagina's van een tweezijdig document (de onevenpagina's zitten altijd rechts):

```
\fancyhf{}
\fancyhf[HEL,HOR]{\thepage}
```

Sommige mensen vinden het mooi om een horizontale lijn in de kop of voettekst te hebben. Deze lijnen kun je aanbrengen door ze een dikte x te geven groter dan nul met de commando's

```
\renewcommand{\headrulewidth}{xcm}

\renewcommand{\footrulewidth}{xcm}
```

Naast een paginanummer is het mogelijk om de titel van het huidige hoofdstuk in de header of footer te vermelden. Je begrijpt dat je daarvoor bepaalde variabelen moet vullen, die bijhouden hoe het huidige hoofdstuk heet. Deze variabelen heten 'markers' en ze worden bij het begin van een hoofdstuk, sectie, etc. gedefinieerd.

Onder bijvoorbeeld een 'chapter' kunnen meerdere 'sections' (en daaronder weer meerdere 'subsections') vallen. Om het mogelijk te maken dat je naar de naam van het hoofdstuk èn de sectienaam kunt verwijzen, houdt LaTeX een marker bij van een hoger en een lager niveau. (Je kunt dus naar de twee hoogste niveau's verwijzen; bij de class 'report' zijn dit 'chapter' en 'section', bij de class 'article' zijn dit 'section' en 'subsection'.)

De markers worden gemaakt met het commando `\markboth{}{}` waarbij het linker argument het hogere niveau indiceert (de variabele waar de informatie in gaat, heet dan ook `\leftmark`) en het rechter argument het lagere (de variabele heet logischerwijs `\rightmark`.

In de praktijk wordt de zaak nog iets gecompliceerd doordat het definiëren van het rechterdeel van `\markboth` vóór het definiëren van het linker deel komt; immers je definieert eerst een hoofdstuk en pas daarna een sectie. Daarom

wordt bij de definitie van het lagere niveau niet `\markboth` aangeroepen maar `\markright{}` (weliswaar verkapt via een commando als `\sectionmark{}`; dit kan nog andere dingen bevatten dan alleen `\markright`). Op die manier hoef je niet de inhoud van `\leftmark` te wijzigen, terwijl dat niet nodig is (sterker nog: het is onmogelijk, omdat je de naam van het hoofdstuk niet voor handen hebt)..

Ik zal een voorbeeld geven om een en ander te verduidelijken. Je preamble kan er bijvoorbeeld zo uitzien:

```
\documentclass[a4paper, twoside]{report}

\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]{%
             \markboth{\MakeUppercase{\chaptername}\
                       \thechapter.\ #1}{}}
\renewcommand{\sectionmark}[1]{%
             \markright{\thesection.\ #1}}
\fancyhf{}
\fancyhf[HEL,HOR]{\textsl{\leftmark}}
\fancyhf[HER,HOL]{\textsl{\rightmark}}
\renewcommand{\headrulewidth}{0.1pt}
\renewcommand{\footrulewidth}{0pt}
\fancyhf[FC]{\thepage}

\begin{document}
```

Dit produceert headers met aan de binnenkant van een dubbelpagina de sectienaam voorafgegaan door het sectienummer, en aan de buitenkant de naam van het hoofdstuk voorafgegaan door 'Hoofdstuk' (de inhoud van `\chaptername`; dit is afhankelijk van de taal) en het hoofdstuknummer. Onder de header staat een dun lijntje.

De footer bestaat enkel uit een paginanummer in het midden van de pagina. Het zal je niet ontgaan dat dit precies is, wat ik voor deze cursus ook gebruik.

(Merk overigens op dat in de inhoud van de headers consequent `\` staat na commando's en punten om te zorgen dat er ruimte tussenkomt en de commando's niet achtereen worden geplakt.)

Epiloog

Wanneer je helemaal tot het einde bent gekomen en het ook allemaal zo'n beetje begrijpt, weet je meer van LaTeX dan menigeen en ben je zeker in staat een prachtig practicumverslag te maken. Bedenk echter wel dat de enige manier om het echt te leren oefenen is. Begin er gewoon eens aan en als het na twee of drie keer nog niet bevalt, dan kun je het alsnog vergeten.

Ik hoop echter dat deze beknopte weergave (die toch best uitgebreid is geworden) voldoende is om je te overtuigen van de kracht van LaTeX. Wanneer je nog meer wil weten of op onverklaarbare wijze vst komt te zitten, dan kun je altijd bij mij (en vele anderen) aankloppen.

Succes!