Gegevensverwerking met δαTEX

Andries Lenstra en Ruud Koning

Samenvatting

De auteurs geven een inleiding in δαTEX. Dat is een verzameling macro's waarmee gegevens eenvoudig kunnen worden verwerkt in een TEX- of LATEX-file. Twee voorbeelden illustreren de grote mogelijkheden van deze macro's. Een uitgebreide handleiding en de macro's zelf staan op `www.xs4all.nl/~rhkoning`.

Keywords: dataverwerking, data-integriteit, δαTEX.

1 Inleiding

TEX produceert zetsel; wie zich met TEX inlaat, probeert vaak al gauw TEX zodanig in te richten, dat het gewenste zetsel met zo weinig mogelijk intypewerk tot stand komt, al is het alleen maar om de kans op fouten te verkleinen. Het macropakket δαTEX, dat op deze bladzijden informeel voorgesteld wordt, is bedoeld als hulpmiddel bij genoemde aanpassing van TEX ten behoeve van al die gevallen waarin in het beoogde zetsel naast 'tekst' ook 'gegevens' te onderscheiden zijn.

Gegevens, of 'data', behoren niet overgetypt te worden maar automatisch uit een centrale bron te worden ingelezen. Bovendien moet zo'n centrale bron gemakkelijk in te richten zijn, zonder dat men zich meteen bij dat inrichten voor allerlei keuzen gesteld ziet waar later niet op terug te komen is. De gegevensopslag onder δαTEX voldoet ten zeerste aan dat laatste adagium, niet het minst omdat de eerste auteur nooit wijs kon worden uit al bestaande data-base programma's; het is echter niet moeilijk δαTEX met bestaande data-bases te laten samenwerken, hoewel nog niet alle code op dat gebied geschreven is. (Momenteel is een windows-programma in ontwikkeling dat gegevens die zijn opgeslagen in een Excel-spreadsheet converteert naar een format dat kan worden verwerkt door δαTEX.) δαTEX haalt de data uit datablokken, en dat zijn (delen van) gewone ascii-files, desnoods deel uitmakend van de onderhavige TEX-file zelf. De bewerkingen die de data vóór het zetten ondergaan, worden verwoord in gewone TEX-code, bijvoorbeeld `\def\Auteur{{\bf \Naam}, \Voorletters}` en zijn dus door de meeste gebruikers eenvoudig te beheren.

In de tweede paragraaf schetsen we de ideeën die aan δαTEX ten grondslag liggen. Dit gebeurt aan de hand van een voorbeeld dat aan de praktijk is ontleend, een referentie-automaat. Dat voorbeeld krijgt een nadere bespreking in de derde paragraaf. De laatste paragraaf is gewijd aan een tweede praktijkvoorbeeld, de vervaardiging van al het zetsel voor een conferentie. Een nauwkeurige gebruiksaanwijzing van δαTEX staat in de Proceedings van de Negende Europese TeX-conferentie in Arnhem, 1995, en op WWW: `www.xs4all.nl/~rhkoning`. Daar staan ook de δαTEX-macro's, deze tekst en alle code die nodig is voor een werkende implementatie van het eerste voorbeeld. De gebruiksaanwijzing bevat precieze aanwijzingen voor de opzet van het systeem, ook bij zodanige hoeveelheden gegevens dat geheugenbeheer belangrijk wordt (δαTEX is in principe tegen zulke situaties opgewassen, want het maakt desgewenst zo weinig control sequences aan, dat het aantal daarvan niet lineair toeneemt met het aantal gegevens in de tekst), en een beschrijving van de 'tools'.

2 Dataverwerking met δαTEX

Veronderstel dat we een wetenschappelijk artikel aan het schrijven zijn en in dat artikel willen refereren aan andere wetenschappelijke artikelen, de 'referenties'. De gegevens daarvan zullen ooit eenmaal ingetypt moeten worden, dus laten we daar mee beginnen. Dat intypen doen we in een aparte file (bijvoorbeeld `artikel.dat`) of gewoon in de TEX-file waar we mee bezig zijn, op een plaats waar TEX inmiddels δαTEX heeft geleerd (`\input datex`). De eerste referentie heet in onze gedachten 'data1', dus als identificerende string die δαTEX graag ziet aan het begin van elke regel met gegevens van 'data1', kiezen we de string `data1`. Aan het werk:

```
**data1* \Titel Data with
**data1* \daTeX; \Jaar 1995;
**data1* \Type \Artikel;\EersteAuteur;
**zzzz*
```

en weer ophouden, want er is een kink in de kabel: hoe vermelden we een auteur? Voornaam of voorletters? Of beide? Moet de affiliatie (werkplek) er ook bij? En staan de voorletters bij de eerste auteur achter de achternaam, of gewoon ervoor?

Kortom, het 'veld' `\EersteAuteur` heeft zelf óók velden. Als we de uiteindelijke verschijningsvorm daarvan nog niet willen vastleggen—en dat klopt: we willen die integendeel juist te allen tijde kunnen veranderen—zullen we er nu al over moeten nadenken hoe we die velden als zodanig herkenbaar maken, om ze vervolgens netjes uniform in te typen—twee dingen die we evenmin willen.

Aan al deze tegenzin kan δαTEX drastisch tegemoet komen; dat ontdekken we als we leren wat δαTEX nu al kan beginnen met de luttele gegevens die we het net hebben verstrekt. Als het bovenstaande datablok door δαTEX is geabsorbeerd (c.q. opgezocht, afhankelijk van de gekozen vorm van geheugenbeheer; datablokken zomaar in de tekst worden altijd geabsorbeerd), is van elk veld apart de inhoud toegankelijk:

```
{\bf het artikel \Of *data1* * ‘\Titel’,
 uit \Jaar}
```

resulteert in het zetsel **het artikel ‘Data with δαTEX’, uit 1995**, enzovoort (waarbij tussen het tweede en derde sterretje in plaats van een spatie de naam van een data-file ingevuld kan worden als het wenselijk is om δαTEX een aanwijzing te geven omtrent de plaats waar het de string die tussen het eerste en tweede sterretje staat, moet zoeken). Hetzelfde zetsel levert de code

```
\def\Artikel{{\bf het artikel ‘\Titel’,
 uit \Jaar}}%
\Of *data1* * \Artikel
```

en daarvan kan de laatste regel vervangen worden door

```
*data1* *
```

omdat `*data1* *` een afkorting is van

```
{\Of *data1* * \Type}
```

en onze eigen opgave van het `\Type` van `data1` juist `\Artikel` luidt. (De gevallen verschillen in de werking van `\Of *data1* *` ; alleen in het tweede geval staat geen `}` die deze opheft.)

Nu weten we dus niet alleen hoe we straks gemakkelijk naar artikelen kunnen verwijzen, maar ook hoe we dat probleem met de invoer van auteursgegevens radicaal oplossen: gewoon alle gegevens die we ooit nodig mochten hebben, apart intypen in een datablok:

```
**Andries* \Naam Lenstra;\Vlts A.J.;
**Andries* \Type \Auteur;
**zzzz*
```

en het in het begin gestaakte intypen zo hervatten:

```
**data1* ... \EersteAuteur *Andries* *;
**zzzz*
```

dus als veldinhoud geen expliciete gegevens, maar een verwijzing daarnaar! (Men zou zulke verwijzingen ‘meta-data’ kunnen noemen; we zagen al een eenvoudiger maar niet minder praktisch voorbeeld van meta-data toen in het veld `\Type` een control sequence werd neergezet.) Met de werkplek gaat het desgewenst net zo:

```
**Andries* \Naam Lenstra;\Vlts A.J.;
**Andries* \Type \Auteur;
**Andries* \Werkplek *UvA* *;
**UvA* \Type \Affiliatie;\Adres Plantage
**UvA*Muidergracht 24;\Pcode 1018 TV;
**UvA* \Plaats Amsterdam; \Naam W.I.N.S.
**UvA*Universiteit van Amsterdam;
**zzzz*
```

Op de mogelijkheden die `*<identificerende string>* *` als veldinhoud biedt, komen we straks terug; eerst geven we de data-typist nog enige aanwijzingen. Het prettige besef, geen woord teveel in te hoeven typen, zal diens ijver ongetwijfeld verdubbelen.

Alle regels die met dezelfde `**<identificerende string>*` beginnen, horen bijeen te staan; als dat nog niet het geval is, is het met vele sorteer-programma’s tot stand te brengen (en als dat wel het geval is, hoeft alleen de eerste `**<string>*` te blijven staan). De regel met `**zzzz*` is precies daar vereist waar een datablok ophoudt, dus waar de data-file eindigt c.q. de tekst verder gaat, of waar het tijd is voor het poneren of opheffen van verstekvelden. Een verstekveld is van toepassing op alles wat volgt, tenzij het ter plaatse terzijde wordt geschoven, namelijk door een expliciet gegeven veld van dezelfde naam, of tezamen met alle andere verstekvelden wordt opgeheven door het commando `\NoDefaults`. Als er vele auteurs achter elkaar moeten worden ingevoerd, hoeft er dus maar één keer te worden meegedeeld dat het gaat om het `\Type \Auteur`; die mededeling geschiedt door de regel

```
\Default \Type \Auteur;
```

Met behulp van dit mechanisme is het extra werk dat de compositie van een identificerende string met zich meebrengt, tot een minimum terug te brengen; in het geval van `\Type \Auteur` ligt het bijvoorbeeld voor de hand om die string gelijk te doen zijn aan de achternaam, en dat—of eigenlijk het omgekeerde—gaat zo:

```
\Default \Naam \Ident;
```

omdat de control sequence `\Ident` steeds de identificerende string geeft. De volgende datablokken illustreren het bovenstaande.

```
\NoDefaults
\Default \Type \Auteur;
\Default \Naam \Ident;
**Andries* \Naam Lenstra; \Vlts A.J.;
**Andries* \Werkplek *UvA* *;
**Andries* \Voornaam \Ident;
**Koning* \Vlts R.H.;\Voornaam Ruud;
**Koning* \Werkplek *VU* *;
**UvA* \Type \Affiliatie;\Adres Plantage
**UvA*Muidergracht 24;\Pcode 1018 TV;
**UvA* \Plaats Amsterdam; \Naam W.I.N.S.
**UvA*Universiteit van Amsterdam;
**VU* \Type \Affiliatie;
**VU* \Naam Vakgroep Econometrie
**VU*Vrije Universiteit;
**VU* \Adres De Boelelaan 1105;
**VU*\Pcode 1081 HV;\Plaats Amsterdam;
**zzzz*
\NoDefaults
\Default \Type \Artikel;
**data1* \Titel Data with \daTeX;
**data1* \Jaar 1995;
**data1* \EersteAuteur *Andries* *;
**zzzz*
```

Om zich voor te stellen wat er zoal binnen bereik komt als `*<een identificerende string>* *` in een veld staat, is maar weinig fantasie vereist. Met de juiste definities van de `\Typen` in kwestie zijn velden van velden van ... velden gemakkelijk toegankelijk:

```
\def\Artikel{{\bf\EersteAuteur}}%
\def\Auteur{\Werkplek}%
\def\Affiliatie{\Plaats}%
*data1* *
```

geeft als zetsel de plaats van de werkplek van de eerste auteur van het artikel `data1`, dus **Amsterdam**. Zo'n faciliteit komt goed van pas in combinatie met het *δα*TEX-commando `\Filter`:

```
\Filter artikel \Type \Artikel
```

bekijkt de file `artikel.dat` van het begin tot het einde en zet de `\Artikel`en die het tegenkomt, neer in de huidige file volgens de implementatie van `\Artikel` die op dat moment van kracht is. (Dit is dus het gereedschap om een 'mail merge' mee te bewerkstelligen.) In de veronderstelling dat alle gegevens inmiddels netjes zijn ingetypt en opgeborgen in aparte data-files, geven we een van de manieren om alléén díe artikelen te selecteren waarvan de eerste auteur zijn werkplek in Amsterdam heeft, onder gebruikmaking van een paar nog niet genoemde stukken gereedschap.

```
\input datex
%
% onderwijst \TeX \daTeX
%
\TheDataFiles
 auteurs (a)
\InSearchOrder
%
% de file auteurs.dat, waarin de
% \Auteurs en de \Affiliaties staan,
% wordt geabsorbeerd (a)
%
% (bij zeer grote data-files moet
% en kan dit anders; zie de gebruiks-
% aanwijzing)
%
\def\Affiliatie{\WrapIn\pLaats\Plaats}%
%
% dit komt neer op:
%  \gdef\pLaats{<inhoud van \Plaats>}
%
\def\Artikel{%
 \def\pLaats{niks}%
%
% geeft \pLaats eerst een verstekwaarde
%
 \def\Auteur{%
  \IfField\Werkplek\Exists
   \Werkplek
  \Fi
 }%
%
% (een \Auteur hoeft geen \Werkplek
% te hebben, maar elke \Affiliatie
% heeft een \Plaats)
%
 \EersteAuteur
%
% definieert eventueel \pLaats opnieuw
%
 \IfCs\pLaats\IsDefinedAs{Amsterdam}%
  \def\Auteur{\Vlts\ \Naam}%
  `\Titel' van \EersteAuteur,
  uit \Jaar\par
%
% anders: niets doen
%
 \Fi
}%
%
\Filter artikel \Type \Artikel
```

Dezelfde werking zou ook te verwezenlijken zijn zonder `\pLaats` te definiëren, met behulp van het commando

```
\IfExistingField\LooksLike
 \Plaats Amsterdam;%
 ...
\Fi
```

In beide gevallen had het niets uitgemaakt als de `\Plaats` óók weer een verwijzing was geweest, dus `\Plaats *Amsterdam* *;` in plaats van `\Plaats Amsterdam;`, behalve dat de voorwaarde dan had moeten luiden

```
\IfCs\pLaats\IsDefinedAs{*Amsterdam* *}%
```

respectievelijk

```
\IfExistingField\LooksLike
 \Plaats *Amsterdam* *;%
```

We besluiten deze inleiding met een geruststelling voor de TEXpert en een kleine opgave aan hetzelfde adres: na `\Of *data1* *` horen natuurlijk alle control sequences die veldnamen voorstellen, de inhoud te geven die bij `data1` hoort. Stel dat er een tweede artikel, `data2`, ingevoerd is, maar zónder `\Jaar`.

Na `\Of *data1* *` is `\Jaar` kennelijk zo gedefinieerd dat TEX uit `{\bf\Jaar}` tenslotte het zetsel **1995** produceert, maar bij de tweede control sequence `\Jaar` in

```
\Of *data1* * \Titel, uit {\bf \Jaar}, en
\Of *data2* * \Titel, uit {\bf \Jaar}
```

hoort TEX níet **1995** te geven—en dat doet het gelukkig ook niet; er komt een foutmelding. Wat weerhoudt TEX? Bedenk dat het níet eerst, zodra het `\Of *data2* *` tegenkomt, een vaste lijst veldnamen op een verstekwaarde mag zetten, want zo'n lijst is er niet. In de oerversie van *δα*TEX deed TEX dat wel, maar het bijhouden van een lijst veldnamen is een onding voor de data-typist.

3 Toepassing 1: een referentie-automaat

Hierboven werd ter inleiding in de werking van δαTEX een referentie-automaat opgezet. Het moge vanzelf spreken dat de manier waarop dat gebeurde, geenszins onder alle omstandigheden de beste hoeft te zijn en dat δαTEX vele andere mogelijkheden ondersteunt. In deze paragraaf wijzen we op een voor de hand liggende vereenvoudiging, ontwerpen we nuttige stukjes δαTEX-gereedschap (lang leve TEX als programmeertaal!) en lichten we een paar onderdelen toe van de code die op het net staat. Tot slot stuiten we op een nieuwe faciliteit van δαTEX door ons voor te stellen dat er heel veel referenties zijn, zodat we zo zuinig mogelijk met het werkgeheugen van het rekentuig om moeten gaan.

De voor de hand liggende vereenvoudiging schuilt natuurlijk in een minder drastische aanpak van de velden van velden. Als op voorhand vast staat dat affiliaties en voornamen nooit vermelding vinden, is het heel goed mogelijk om van het `\Type \Auteur` af te zien en de `\Auteurs` op te geven als, bijvoorbeeld,

```
**data1* \Auteurs Lenstra, A.J.,
**data1*<andere auteurs>, Koning, R.H.;
```

TEX zoekt dan zelf uit hoeveel auteurs er zijn, en bij welk rangnummer de gegeven achternamen en voorletters horen. Veel eenvoudiger te implementeren is het om steeds maar één auteur tegelijk op te geven, net als in paragraaf 2. Bij vier auteurs zou dat zó kunnen:

```
**data1* \EersteAuteur Lenstra, A.J.;
**data1* \EersteTussenauteur ..., ...;
**data1* \TweedeTussenauteur ..., ...;
**data1* \LaatsteAuteur Koning, R.H.;
```

Als er altijd na de komma vóór de voorletters een spatie staat, geven de volgende `\NaamEersteAuteur` en `\VltsEersteAuteur` wat ze beloven:

```
\let\ea\expandafter
%
% (doet \daTeX zelf al)
%
\def\VoorDeKomma#1, #2+{#1}%
\def\NaDeKomma#1, #2+{#2}%
\def\NaamEersteAuteur{%
 \WrapIn\eErsteAuteur\EersteAuteur
%% of:
% \edef\eErsteAuteur{\EersteAuteur}%
 \ea\VoorDeKomma\eErsteAuteur+}%
\def\VltsEersteAuteur{%
 \WrapIn\eErsteAuteur\EersteAuteur
%% of:
% \edef\eErsteAuteur{\EersteAuteur}%
 \ea\NaDeKomma\eErsteAuteur+}%
```

De code die op het net staat, hanteert wél het `\Type \Auteur`, zoals uiteengezet in de tweede paragraaf, maar niet het `\Type\Affiliatie`, omdat de behoefte aan de vermelding van een `\Werkplek` zich nog niet heeft voorgedaan. Dat veld toevoegen kan altijd nog.

Met een apart `\Type \Auteur`, en elke auteursvermelding in de vorm `*Andries* *`, hoeven we, om te voldoen aan de vaak gestelde eis dat alleen van de eerste auteur de `\Naam` aan de `\Vlts` voorafgaat, in de definitie van `\Artikel` alleen de definitie van `\Auteur` te veranderen:

```
\def\Artikel{...
 \def\Auteur{\Kap\Naam, \Vlts}%
 \EersteAuteur
 \def\Auteur{\Vlts{} \Naam}%
 <eventuele overige auteurs>
 ...
}%
```

`\Kap` zorgt ervoor dat de lezer gruwelen als **het artikel van de Boer [1993]** bespaard blijven, door het erop volgende token helemaal te expanderen en de eerste letter van het resultaat kapitaal te zetten, dus als hoofdletter. Een werkende implementatie is:

```
\let\ea\expandafter
\def\Kap#1{\edef\ExpToken{#1}%
 \ea\Hap\ExpToken}%
\def\Hap#1{\uppercase{#1}}%
```

en met `\def\dH{de heer}` produceert

```
**de Boer* \Naam \Ident;
**de Boer* \Type \Auteur;\Titel \dH;
**zzzz*
\def\Auteur{{\bf\Kap\Titel\ \Kap\Naam}}%
*de Boer* *
```

het zetsel **De heer De Boer**; ook het systeem dat in de vierde paragraaf ter sprake komt, kan het niet stellen zonder de diensten van een commando als `\Kap`.

De laatste opmerking die we over de code op het net willen maken, betreft de verschillende manieren waarop δαTEX weet te achterhalen of het de eerste keer is dat naar een bepaalde referentie verwezen wordt. (δαTEX wil namelijk in dat geval álle auteurs vermelden, en niet volstaan met de eerste auteur 'et al.'.) Kort en snel is de code die berust op de eigenaardigheid van TEX, waar onlangs weer door Paul Taylor op is gewezen, dat een control sequence die gevormd wordt met behulp van `\csname` en `\endcsname` equivalent is aan `\relax` zolang hij nog niet met `\def` een betekenis heeft gekregen. `\Ident` leent zich bij uitstek voor de vorming van zo'n control sequence; als deze meteen na de toets die hem gelijk bevindt aan `\relax`, ge`\gdef`inieerd wordt als willekeurig wat (`\global` omdat alles zich in een groep afspeelt), zal hij alle volgende keren ongelijk zijn aan `\relax`.

Handig, maar wel een belasting voor het werkgeheugen; elke eerste verwijzing stopt daar zo immers een nieuwe control sequence in. Als het werkgeheugen voor die belasting te klein is, moet het schijfgeheugen te hulp komen: δαTEX dient dan de identificerende strings die het ontmoet, naar een file weg te schrijven, en die file te inspecteren als het wil weten of het een bepaalde string al eerder is tegengekomen. Dit is met weinig moeite voor elkaar te krijgen; met iets meer moeite is zelfs te bereiken—en

daar is natuurlijk geen geheugengebrek voor nodig—dat de data-file van `\Artikel`en met de nieuwe verwijsgegevens wordt ververst. In de uiteindelijke lijst referenties kunnen dan bij elk artikel de nummers staan van de bladzijden waarop naar dat artikel wordt verwezen, waardoor die lijst veel bruikbaarder is dan men meestal aantreft.

Het enige wat hiervoor nodig is, is ervoor te zorgen dat het wegschrijven, waartoe de opdracht in `\Artikel` staat, gebeurt in het format van een regel van een δαTEX datafile, met na de identificerende string nog een veld waarvan de inhoud het paginanummer van de verwijzing karakteriseert, en de naam het rangnummer daarvan plus het artikel waarin verwezen wordt. Stel dat dat `data4` heet en dat daarin op bladzijde 717 voor de negenentwintigste maal `*data1* *` staat, dus verwezen wordt naar `data1`, dan zou de op die plaats weggeschreven regel er zo uit kunnen zien:

```
**data1* \VerwijzingNrXXIXinDataIV p. 717;
```

De inspectie op strings die al geweest zijn, is eenvoudig te verwezenlijken door de wegschrijf-file te `\Filteren` in de staat waarin die zich op dat moment bevindt. (Als δαTEX gaat opzoeken in plaats van filteren, dan denkt het dat het klaar is zo gauw het de eerste regel gevonden heeft, terwijl het met `\Filter` de hele file doorloopt.) Om de data-file te verversen hoeven we alleen maar de regels van de volledige wegschrijf-file op de juiste plaatsen in de oorspronkelijke file `artikel.dat` in te voegen. Geschikt `\Filteren` van het resultaat levert tenslotte de lijst referenties volgens de beloofde specificatie.

4 Toepassing 2: een programmaboekje

De Vereniging voor Statistiek en Operationele Research (VVS) organiseert jaarlijks de zogenaamde Statistische Dag. Op deze dag worden 's ochtends twee hoofdlezingen gegeven en 's middags verschillende presentaties in parallelsessies. Het programma wordt gedrukt in een programmaboekje, dat naast het programma ook samenvattingen van de voordrachten en enige advertenties bevat. Voor de samensteller van het boekje kan de uiteindelijke productie ontaarden in buitengewoon veel en onaangenaam werk: hij (of zij) moet ongeveer twintig sprekers benaderen, van alle sprekers een titel van de lezing opnemen in het programma, en twintig samenvattingen laten afdrukken in het boekje. Gegevensintegriteit is in dit geval buitengewoon belangrijk: het is pijnlijk als een spreker aangeschreven wordt als 'Ruud Koning', in het programma vermeld staat als 'drs Ruud H. Koning' en volgens de samenvatting 'dr. R.H. Koning' heet. Verder staat het slordig als de ene spreker met een voornaam wordt aangeduid en een andere spreker met initialen. Daarnaast zijn fouten in de titelaanduiding snel gemaakt (schrijven we 'prof. dr.', 'prof.dr' of 'prof.dr.'). Veel van dit soort problemen kunnen worden voorkomen door δαTEX te gebruiken. Het schrijven van brieven en rekeningen, ook aan de adverteerders, en het maken van een programmaboekje kunnen hiermee in hoge mate worden geautomatiseerd, en dat betekent dat de organisatie zich niet bezig hoeft te houden met typewerk, maar zich kan richten op het organiseren van de dag zelf. Als het verder niet veel werk is om een mailing te versturen aan de sprekers, is er geen reden om de samenvatting en personalia (die vaak niet rechtstreeks van de spreker afkomstig zijn en soms handgeschreven terechtkomen bij de organisator) niet door de sprekers te laten corrigeren, of om uitgenodigde sprekers na de Statistische Dag niet schriftelijk te bedanken voor hun voordracht.

In deze toepassing worden twee bestanden met gegevens gebruikt: een bestand met lezingen (`lezing.dat`) en een bestand met personen (`spreker.dat`). Een lezing heeft de volgende datavelden: `\titel` met de titel, `\spreker` met de spreker en `\samenvatting` met de filenaam waarin de samenvatting staat. Een voorbeeld is[1]

```
**hfd1* \titel Vijftig jaar VVS: nog%
**hfd1* eenmaal de magere jongelieden;
**hfd1* \spreker *zwet* *;
**hfd1* \samenvatting vzwet.tex;
**hfd1* \Type\lezing;
```

In het bestand `spreker.dat` komen de volgende regels voor:

```
**zwet* \naam van Zwet;\titels \profdr;
**zwet* \vlts W.R.;\Type \persoon;
**zwet* \vakgroep Vakgroep Mathematische%
**zwet* Statistiek;\pcode 2333 CA;
**zwet* \straat Niels Bohrweg 1;
**zwet* \plaats Leiden;
```

Zoals eerder betoogd is het niet noodzakelijk om de verschillende gegevenstypen (lezingen en personen) in verschillende bestanden op te slaan. Met behulp van deze gegevens zijn we in staat om bepaalde elementen uit het programmaboekje te maken. Allereerst kondigen we de lezingen aan in het programma zelf. Later moet de samenvatting van de lezing worden afgedrukt. De aankondiging in het programma bestaat uit de titel en de spreker.

De volgende macro's zorgen ervoor dat een lezing wordt afgedrukt:

```
\newcommand{\profdr}{prof. dr. }
\newcommand{\Spreker}{}
\newcommand{\Titels}{}
\newcommand{\Vlts}{\Kap}
\newcommand{\Titel}{}

\newcommand{\persoon}{%
 \renewcommand{\Titels}{}%
 \IfField\titels\Exists
  \renewcommand{\Titels}{\titels}%
 \Fi
 \renewcommand{\Vlts}{\Kap}%
 \IfField\vlts\Exists
  \renewcommand{\Vlts}{\vlts\ }%
```

[1] De voorbeelden zijn gebaseerd op het programma van het jubileumcongres van de VVS op 5 en 6 april 1995 en zijn enigszins vereenvoudigd.

```
 \Fi
 \Titels\Vlts\naam
}

\newcommand{\lezing}{%
 \renewcommand{\Titel}{titel volgt}%
 \IfField\titel\Exists
  \renewcommand{\Titel}{‘\titel’}%
 \Fi
 \renewcommand{\Spreker}{spreker onbekend}%
 \IfField\spreker\Exists
  \renewcommand{\Spreker}{\spreker}%
 \Fi
 {\bf \Titel, \Spreker}%
}
```

Als de tekst van het programmaboekje `*hfd1* *` bevat, resulteert dit in het zetsel **‘Vijftig jaar VVS: nog eenmaal de magere jongelieden’, prof. dr. W.R. van Zwet**. Als we kijken naar de definitie van de macro `\lezing` is duidelijk wat er gebeurt. Elke macro van het type `\lezing` voert uiteindelijk tot de expansie van de macro’s `\Titel` en `\Spreker`. De macro `\Titel` expandeert tot `\titel`, mits dat veld bestaat. Als dat veld niet bestaat, expandeert `\Titel` tot **titel volgt**. De titel is in ons geval bekend, dus `\Titel` expandeert tot **Vijftig jaar VVS: nog eenmaal de magere jongelieden**. De veldnamen kunnen zo eenvoudig verstekwaarden krijgen. Evenzo expandeert de macro `\Spreker` tot `\spreker`. Echter, de macro `\spreker` expandeert niet tot tekst, maar tot `*zwet* *`. Dit is een verwijzing die onder meer de expansie van `\persoon` behelst. Interessant aan de expansie van `\persoon` is dat `\Titels` expandeert tot `\profdr`, dat gedefinieerd is als `prof. dr. `. Op deze wijze is eenduidigheid in titulatuur ook eenvoudig te bewerkstelligen. Dit veld kunnen we eveneens gebruiken bij de adressering van een brief:

```
\newcommand{\Aanhef}{}
\IfField\titels\Exists
 \IfExistingField\LooksLike
   \titels\profdr;%
  \renewcommand{\Aanhef}{{\bf%
    Aan de hooggeleerde heer
    \titels\Vlts\naam}}%
 \Fi
\Fi
```

Hier vergelijkt δαTEX de inhoud van het veld `\titels` met `\profdr` zonder `\profdr` te expanderen. In ons voorbeeld zal `\Aanhef` uiteindelijk expanderen tot **Aan de hooggeleerde heer prof. dr. W.R. van Zwet**. Uiteraard kan een vergelijkbare macro worden gebruikt om te controleren of een brief wordt gericht aan een vrouw, een man, of een man of een vrouw.

De verwijzing `*hfd1* *` levert dus uiteindelijk **‘Vijftig jaar VVS: nog eenmaal de magere jongelieden’, prof. dr. W.R. van Zwet**. In de LATEX-code van het programmaboekje wordt alleen `*hfd1* *` gebruikt, en niet de ‘harde’ tekst waarin `*hfd1* *` tenslotte resulteert. Dit heeft als groot voordeel dat wijzigingen in de gegevens slechts één keer ingevoerd hoeven te worden om in het uiteindelijke zetwerk alle keren te worden doorgevoerd.

Naast het programma bevat het programmaboekje ook de samenvattingen van de voordrachten. Bij de samenvattingen wordt niet alleen een korte beschrijving van de voordracht afgedrukt, maar ook uitgebreide adresgegevens van de spreker. De expansie van `\lezing` moet zó worden gewijzigd dat het bestand met de tekst van de samenvatting wordt ingelezen, en de expansie van `\persoon` gaat nu ook de adresgegevens afdrukken.

```
\renewcommand{\persoon}{%
 \renewcommand{\Titels}{}%
 \IfField\titels\Exists
  \renewcommand{\Titels}{\titels}%
 \Fi
 ...
 \renewcommand{\Email}{}%
 \IfField\email\Exists
  \renewcommand{\Email}{ {\texttt{\email}}}%
 \Fi
 \Titels\Vlts\naam, \Vakgroep
 \Straat\Pcode\Plaats\Email
}%
\renewcommand{\lezing}{%
 \renewcommand{\Titel}{titel volgt}%
 \IfField\titel\Exists
  \renewcommand{\Titel}{\titel}%
 \Fi
 \renewcommand{\Spreker}{spreker onbekend}%
 \IfField\spreker\Exists
  \renewcommand{\Spreker}{\spreker}%
 \Fi
 \IfField\tweedeauteur\Exists
  \renewcommand{\Spreker}{\spreker,
   \tweedeauteur}%
 \Fi
 \renewcommand{\Samenvatting}{Er is helaas
  geen samenvatting beschikbaar}%
 \IfField\samenvatting\Exists
  \renewcommand{\Samenvatting}{%
   \input abstract/\samenvatting}%
 \Fi
 {\bf ‘\Titel’, \Spreker}\smallskip
 \Samenvatting\bigskip
}
```

De uiteindelijke lijst met samenvattingen genereren we met het commando `\Filter lezing\Type\lezing`. Alle `\lezing`en in `lezing.dat` expanderen nu volgens de bovenstaande definitie. Als alle gegevens in `lezing.dat` zijn bijgewerkt, zal van geen enkele lezing de samenvatting worden vergeten. Indien van een lezing geen samenvatting beschikbaar is, wordt een tekst van die strekking afgedrukt. Als het veld `\samenvatting` wel bestaat, zoekt δαTEX het aangegeven bestand in de subdirectory `abstract` en leest het bestand in. De tekst in dat bestand moet natuurlijk wel correcte LATEX-code zijn. Merk op dat `\persoon` nu ook de adresgegevens van de spreker (of een spreker en zijn of haar co-auteurs) afdrukt. Voor elke expansie van `\persoon` wordt nagegaan of alle relevante velden beschikbaar zijn. Er wordt niets (of een verstektekst) afgedrukt als dat niet het geval is. In ons voorbeeld is het resultaat van `hfd1` nu

‘Vijftig jaar VVS: nog eenmaal de magere jongelieden’, prof. dr. W.R. van Zwet, Vakgroep Mathematische Statistiek, Niels Bohrweg 1, 2333 CA Leiden.

<zetsel abstract/vzwet.tex>

Uit het bovenstaande kan de indruk ontstaan dat het samenstellen van een programmaboekje niet veel eenvoudiger wordt door het gebruik van δαTEX. In plaats van het zetten van teksten zou men nu veel tijd kwijt zijn met het schrijven van macro’s. Dit is een misvatting: een beperkt aantal macro’s genereert een compleet programmaboekje. Juist door het scheiden van gegevens en tekst wordt grote consistentie in het eindresultaat bereikt en is het eenvoudig om lezingen die één voor één worden aangemeld, bij de organisatie te verwerken. Het uiteindelijke zetwerk wordt door LATEX en δαTEX gedaan; met een zeer beperkt aantal macro’s en regels tekst is in een handomdraai een programmaboekje gegenereerd (de LATEX-file voor het programma van het jubileumcongres van de VVS bestond uit zo’n 300 regels, de uiteindelijke omvang van het programmaboekje was 44 pagina’s). Het uiteindelijke werk is niet het typen van het boekje, maar het vullen van de bestanden `lezing.dat` en `spreker.dat`.