Nybörjarguide i LaTeX, första upplagan.

Att TeXa: en praktisk guide

Simon Sigurdhsson

Sammanfattning En inkomplett guide till att skriva och typsätta LaTeX-dokument riktad till studenter på Chalmers Tekniska Högskola, specifikt programmen Teknisk Matematik och Teknisk Fysik. Inspiration har tagits från bland annat von Schultz (2005) och Voß (2010), men främst från Oetiker et al. (2011).

Tack till

Phaddergrupp 255,
som korrekturläst och
hjälpt till att förbättra boken

och

Christian von Schultz,
Tobias Oetiker och *Herbert Voß*
för inspirerande texter på samma tema.

Göteborg, 2011

Innehåll

Inledning **1**
Vad är LaTeX? . . . 1
Varför LaTeX? . . . 1
I denna introduktion . . . 2

I Grundläggande begrepp **5**
LaTeX-dokument . . . 5
Dokumentets layout . . . 7
Från TeX till PDF . . . 9
Filer du kanske stöter på . . . 10

II Typsättning med LaTeX **13**
Text- och språkstruktur . . . 13
Figurer, tabeller, listor . . . 18
Typografiska betänkligheter . . . 23
Andra viktiga delar . . . 25

III Matematik med LaTeX och AMS **29**
Att visa ekvationer . . . 29
En snabbkurs i LaTeX-matematik . . . 30
Ekvationsomgivningar . . . 38
Mellanrum . . . 42
Punkter . . . 42
Enheter med siunitx . . . 44

IV Grafik med LaTeX **45**
Inkludera grafik med graphicx 45
Rita med PGF/TikZ 46

V Referenser med BibTeX **49**
BibTeX-databasen 49
BibTeX med LaTeX 51

VI Vidare läsning **55**
Andra resurser 55
Tips för stora projekt 57
Rekommenderade paket 58
Andra TeX-baserade projekt 62

Referenser **65**

A En enkel mall **67**

Inledning

Att publicera något, oavsett om det är långa böcker eller korta artiklar, är inte helt enkelt. Förr, innan det fanns en dator i varje hem och innan programvaror som Microsoft Word och OpenOffice blev tillgängliga för var man så var publicering något komplext, något som utfördes av många olika personer. Arbetet delades upp i olika bitar och en expert inom varje område fick sköta den biten; författande, typsättning, design och så vidare.

Moderna programvaror (ordbehandlare, främst) fungerar på ett helt annat sätt. De ger all makt till författaren, som kanske varken vill eller bör (författaren är ju inte typsättare) ha makt över typografin. Som en konsekvens blir typsättningen av dokument inte alltid bra — användarna är under illusionen att ett estetiskt tilltalande dokument även är ett bra dokument rent typografiskt. Så är givetvis inte fallet.

För att få snygga dokument måste man återskapa den gamla ordningen där varje uppgift löses av någon som är bra på det. En författare ska inte behöva berätta *hur* saker ska se ut — det ska designern göra — författaren ska bara berätta *vad* saker är[1]. LaTeX låter dig som författare göra just detta.

Vad är LaTeX?

LaTeX (uttalas [la'tɛʃ]) är i princip en påbyggnad till TeX, ett typsättningssystem. Man kan se det som att LaTeX är designern, och TeX är typsättaren. LaTeX berättar i någon mening för TeX hur den tror att du vill ha ditt dokument (utifrån dokumentklassen samt den kod du skriver) och låter sedan TeX typsätta detta för att skapa ett ”tryckt” dokument.

Men LaTeX är inte bara ett typsättningssystem; både språket dokumenten skrivs i och den kompilator som omvandlar dokumenten till DVI-filer kallas LaTeX (men just kompilatorn brukar benämnas `latex`, eftersom det är så den körs från terminalen). Det finns även modernare versioner av LaTeX-kompilatorn, till exempel pdfLaTeX som skapar PDF-dokument direkt (så att man slipper konvertera dem med `dvipdf`) och XeLaTeX, som baseras helt på Unicode och har bättre stöd för diverse typsnitt.

Varför LaTeX?

Fördelen med LaTeX är att författaren endast behöver lära sig några enkla kommandon — sådana som gör texten kursiv eller infogar en figur — men att dokumentet ändå håller mycket hög standard, precis som om det typsatts av en riktig typsättare.

[1] Detta är inte helt olikt den uppdelning som görs mellan HTML och CSS.

Eftersom språket man skriver dokument i uppmanar till struktur och ordning blir det dessutom lättare att skriva strukturerade texter. Även om saker som finns i en vanlig ordbehandlare (stavningskontroll, till exempel) saknas i själva språket så kan (och bör) dessa tillhandahållas av externa program, vilket gör att LaTeX kan koncentreras på att vara bra på *en* sak: att typsätta dokument. Saker som är komplicerade eller omöjliga att göra i en vanlig ordbehandlare, till exempel korsreferenser, figurer, tabeller, fotnoter, referenslistor och dylikt är mycket enkla att göra i LaTeX och oftast fullt automatiserade.

Dessutom har LaTeX traditionellt använts av akademiker eftersom det gör typsättning av just matematik och fysik både enkelt och visuellt attraktivt.

I denna introduktion

Den här korta introduktionen kommer att visa dig hur man på ett enkelt sätt typsätter dokument med LaTeX i vanliga tillämpningar. Dessutom kommer den framåt slutet peka på specifika paket eller resurser som kan vara användbara för mer avancerade fall.

Efter att ha läst den här introduktionen bör läsaren kunna skriva dokument och rapporter utan problem. Det är dock inte tänkt att denna introduktion ska vara en fullgod referens till LaTeX; för detta rekommenderas istället Lamport (1994) och Mittelbach et al. (2004).

Introduktionen innehåller följande delar:

Del I, Grundläggande begrepp beskriver den grundläggande strukturen hos LaTeX-dokument och hur det språk dokumenten skrivs i fungerar i korta drag. Efter denna del bör du veta på ett ungefär hur LaTeX fungerar.

Del II, Typsättning med pdfLaTeX beskriver i detalj hur man skriver ett vanligt LaTeX-dokument, och förklarar några av de viktigaste miljöerna och kommandona som används. Efter denna del bör du kunna skriva enkla dokument med LaTeX.

Del III, Matematik med LaTeX och AMS beskriver hur man på bästa sätt använder LaTeX tillsammans med AMS-paketen för att typsätta det LaTeX typsätter bäst, matematik, och går även kort in på hur man typsätter en del fysik med paketet siunitx.

Del IV, Grafik med LaTeX beskriver hur man inkluderar grafik i LaTeX med paketet graphicx, och visar några korta exempel på hur man kan rita direkt i LaTeX med PGF/TikZ. Efter den här och föregående del bör du kunna skriva fullständiga rapporter med LaTeX.

Del V, Referenser med BibTeX beskriver hur du använder BibTeX för att hålla koll på och använda referenser i LaTeX, och beskriver även i korthet paketet chscite som hjälper dig att typsätta referenser på det sätt Chalmers bibliotek rekommenderar. Efter denna delen bör du kunna skriva långa arbeten (till exempel kandidatrapporter) i LaTeX.

Del VI, Vidare läsning tipsar om andra resurser, paket, dokumentklasser och rekommendationer som kan vara av nytta när du skriver långa (eller korta) rapporter. Kan vara en språngbräda om du vill göra något som inte förklaras i resten av introduktionen.

Appendix A, En enkel mall innehåller en mall du med fördel kan basera dina framtida LaTeX-dokument på. Den är fullt kommenterad och motiverar de paket som inkluderas och kommandon som definieras.

Det är viktigt att läsa delarna i rätt ordning; varje del bygger på de föregående, och de är ju trots allt inte särskilt långa. Se till att studera och förstå de exempel som presenteras, och lek gärna lite själv om du inte riktigt förstår. Det finns inget bättre sätt att lära sig än att vara nyfiken!

LaTeX finns till många plattformar, och finns installerat på Chalmers Linuxdatorer. Vill du installera LaTeX på din egen dator finns det sannolikt i din pakethanterare (om du använder Linux), alternativt i form av TeX Live[1]. Använder du Mac OS X finns det istället MacTeX[2], och till Windows finns MiKTeX[3]. Den här introduktionen kan tyvärr inte ge fullständiga instruktioner för att installera dessa paket (det är inte introduktionens syfte); konsultera istället respektive pakets dokumentation.

Under introduktionens gång kommer det refereras till så kallade *paket*, som används för att utöka LaTeX med intressanta (och ibland nödvändiga) funktioner. Dessa paket kommer oftast att beskrivas lite kort, men vill man se fullständig dokumentation för varje paket kan man leta på *the Comprehensive TeX Archive Network* (CTAN)[4]. Det lättaste sättet att hitta paket på CTAN är att använda dess sökfunktion[5].

[1] http://www.tug.org/texlive/ [2] http://www.tug.org/mactex/ [3] http://miktex.org/ [4] http://www.ctan.org/
[5] http://www.ctan.org/search/

I
Grundläggande begrepp

Introduktionen har förklarat lite kort vad TEX/LATEX är och varför du bör använda systemet för att typsätta dina rapporter, artiklar och böcker. Denna del kommer att inleda din TEX-bana genom att lite kort förklara hur ett LATEX-dokument är uppbyggt, några viktiga begrepp samt hur LATEX-kompilatorn (i det här fallet `pdflatex`) fungerar.

LATEX-dokument

De dokument LATEX läser in är enkla textfiler, oftast med filändelsen `.tex`. Dessa kan skapas med vilken textredigerare som helst, men det rekommenderas att man använder en textredigerare med syntaxfärgning (då det underlättar felsökande) och stavningskontroll. Oftast sparar man filen med teckenkodningen **UTF-8** (detta gör de flesta moderna textredigerare), men det finns en risk att filerna sparas med teckenkodningen **ISO-8559-1**[1], och då måste man ha koll på detta eftersom man måste berätta för LATEX hur filen ska läsas in.

Tomrum

Tomrumstecken, det vill säga mellanslag, tabbar och liknande behandlas alla som ”tomrum” av LATEX. Flera sådana tecken i rad tolkas som ett enda tomrum, vilket gör att man kan ex. indentera textstycken i sin LATEX-fil utan att detta påverkar resultatet. Tomrum i början av en rad ignoreras generellt sett, och ett enkelt nyradstecken tolkas också som tomrum.

Två nyradstecken på rad (det vill säga en tom rad mellan två textrader) tolkas som styckesindelning, och flera tomma rader efter varandra tolkas som en enda tom rad. Detta gör att LATEX-filerna blir mycket enkla att både skriva och läsa, även om man inte är en särskilt bevandrad TEXniker.

Specialtecken

Vissa tecken är speciella för LATEX. Dessa används internt av LATEX för att representera speciella konstruktioner, och kan inte användas hur som helst i ett LATEX-dokument. Gör man det ändå slutar det vanligtvis bara med att tecknet inte syns, men det kan potentiellt göra att ditt dokument inte ens kompilerar. Specialtecknen är inte särskilt många:

```
# ^ & _ { } ~ \ % $
```

[1] Det finns även andra teckenkodningar som kan dyka upp; Windows-1252 är en sådan.

För att använda dem måste man lägga till ett (bakvänt) snedstreck, \, innan tecknet man vill använda (dock ej innan \ självt, eftersom LaTeX tolkar \\ som att man vill tvinga fram en radbrytning — vill man ha ett \ använder man istället `\textbackslash`). När det gäller ^ och ~ måste man dessutom lägga till måsvingar efteråt, eftersom dessa tecken normalt används för att modifiera vokaler. Tecknen måste alltså skrivas på följande sätt:

```
\# \^{} \& \_ \{ \} \~{} \textbackslash \% \$
```

LaTeX-kommandon

För att säga åt LaTeX vad som ska göras används kommandon. Dessa kommandon följer några enkla regler:

- De börjar med ett bakvänd snedstreck (\) och följs av sitt namn, som bara får innehålla bokstäver[1]. Kommandon avslutas av ett mellanslag, en siffra eller annan godtycklig ”icke-bokstav”.
- De är skriftlägeskänsliga (`\Kommando` är inte samma sak som `\kommando`).
- Vissa kommandon har även en ”stjärnvariant”, då en stjärna (∗) läggs till på slutet av kommandonamnet.

LaTeX ignorerar tomrum efter ett kommando. Det är därför nödvändigt, om man vill ha ett mellanslag efter ett kommando, att lägga till antingen en tom parameter {} och ett mellanslag eller ett speciellt mellanslagskommando (till exempel ~) mellan kommandot och nästa ord. Detta stoppar LaTeX från att äta upp allt tomrum efter kommandot.

Vissa kommandon kräver en obligatorisk och/eller frivillig parameter som på ett eller annat sätt bestämmer hur kommandot beter sig. Den generella strukturen hos ett kommando med sådana parametrar är relativt enkel:

```
\kommando[frivillig parameter]{obligatorisk parameter}
```

Finns det ingen obligatorisk parameter, eller om den frivilliga parametern inte används, kan hela biten inklusive hakparantes/måsvinge helt utelämnas. Ibland kan den frivilliga parametern innehålla många bitar information, till exempel olika flaggor till paket. Dessa brukar då separeras med ett kommatecken.

Kommentarer

Ibland kan det vara bra att kommentera bort (alltså säga åt LaTeX att ignorera) vissa bitar av dokumentet. Det kan vara för att förklara olika kommandon eller definitioner, eller för att ta bort ett stycke man inte är helt säker på. Detta görs med procenttecknet; när LaTeX stöter på ett sådant (såvida det inte sitter ett \ före) så ignorerar den hela resten av den raden i TeX-filen:

```
Lite text\ldots{} % ...och en kommentar
```

[1] Det finns även ett antal kommandon som består av \ och exakt en icke-bokstav, till exempel `\&`.

Grundläggande struktur

När du nu vet hur LaTeX tolkar tomrum, vad ett kommando är och hur du kommenterar delar av dina filer är det dags att förklara lite närmre hur en typisk LaTeX-fil ser ut. Kompilatorn förväntar sig en viss struktur; till exempel så måste varje dokument inledas med kommandot

```
\documentclass{...}
```

som berättar för LaTeX vilken dokumentstil du vill använda (det finns ett antal olika, för till exempel rapporter, böcker eller brev).

Därefter bör man köra kommandon som påverkar hela dokumentet, till exempel genom att ladda in extra paket eller definiera nya kommandon. Paket laddar man in med kommandot `\usepackage`:

```
\usepackage{...}
```

Både `\documentclass` och `\usepackage` tar en obligatorisk parameter (namnet på stilen/paketet) och en frivillig parameter som används för att skicka inställningar till stilen eller paketet som laddas in — en närmare förklaring av båda dessa kommandon kommer senare.

När allt förarbete är gjort kan man inleda dokumentet. Först berättar man för LaTeX att innehållet börjar med hjälp av kommandot

```
\begin{document}
```

och därefter följer dokumentets innehåll. När allt innehåll är slut avslutar man dokumentet med det snarlika kommandot

```
\end{document}
```

som säger åt LaTeX att det inte finns något mer som ska typsättas. Vad man egentligen gör här är att skapa en omgivning med namnet `document`, som betecknar att detta är dokumentets innehåll. Omgivningar förklaras utförligt på sidan 18.

Dokumentets layout

LaTeX är inte helt oflexibelt, och man kan med hjälp av olika kommandon i inledningen till dokumentet ändra både utseendet och beteendet hos detsamma. Utseendet ändrar man med så kallade dokumentklasser (varje dokument måste specificera exakt en sådan), medan extra funktionalitet läggs till i form av så kallade paket.

Dokumentklasser

Den första informationen LaTeX behöver när den kompilerar ett dokument är vilken typ av dokument författaren vill skapa. Detta specificeras med hjälp av kommandot `\documentclass`:

```
\documentclass[inställningar]{klass}
```

Här berättar `klass` vilken sort dokument som ska skapas, eller vilken stil som ska användas. Med de flesta LaTeX-distributioner följer ett antal standardklasser, och av dessa är det fyra man kan tänkas komma i kontakt med relativt ofta:

article är en klass för artiklar till tidsskrifter, korta rapporter, dokumentation, och annat som inte har någon annan, specifik klass. Om du inte vet vilken klass du vill ha, börja med article.

report är en klass för längre rapporter (sådana med flera delar eller kapitel), kortare böcker, kandidatrapporter, examensarbeten och doktorsavhandlingar.

book är en klass för riktiga böcker, som ska gå till tryck.

beamer är en klass för presentationer och overhead-bilder.

Utöver dessa finns det även mer esoteriska klasser så som memoir, koma-script, octavo och refman. Den intresserade uppmanas att leta upp dokumentationen för dessa på CTAN.

Standardklasserna har också ett antal inställningar som kan ges som en kommaseparerad lista i den frivilliga parametern till `\documentclass`:

`10pt,11pt,12pt` sätter textstorleken (för brödtext) i dokumentet. Om ingen av dessa anges är `10pt` standard.

`a4paper,…` definierar pappersstorleken. Det finns ett antal olika, bland annat `a5paper` och `b5paper`, men standard är `letterpaper` — se därför till att ändra, eftersom denna pappersstorlek inte används i Sverige.

`onecolumn,twocolumn` sätter antalet kolumner i dokumentet. Standard är en kolumn (`onecolumn`), och vill man ha två kolumner rekommenderas det att man använder paketet multicol istället för alternativet `twocolumn`.

`twoside,oneside` specificerar om man vill ha ett dubbel- eller enkelsidigt dokument. article och report är enkelsidiga och book är dubbelsidig om inget annat anges. Notera att detta bara påverkar dokumentets stil; `twoside` berättar *inte* för din skrivare att du vill ha dubbelsidiga utskrifter.

Säg till exempel att du vill skriva en rapport och att du vill använda den allmänt accepterade textstorleken 12 pt på ett A4-papper. Det är ingen lång rapport, kanske till och med en inlämningsuppgift, och du bestämmer dig därför för att använda article-klassen:

```
\documentclass[12pt,a4paper]{article}
```

Paket

När du skriver ett dokument kommer du troligtvis att märka att vissa saker är svåra eller ointuitiva (kanske rent av omöjliga[1]) att göra med grundläggande LaTeX-kod. Lyckligvis finns det då nästan alltid ett paket på CTAN som förenklar det du vill göra. Installerade paket kan inkluderas med kommandot `\usepackage`:

```
\usepackage[inställningar]{paketnamn}
```

Notera dock att du måste installera dessa paket innan du kan inkludera dem i ditt LaTeX-dokument. Din LaTeX-distribution kommer förmodligen med de flesta paket förinstallerade, men om du får

[1] Tekniskt sett är de inte omöjliga, eftersom TeX är Turingkomplett. Nästan alla paket är implementerade med LaTeX-kod.

ett felmeddelande när du försöker använda ett paket beror det troligtvis på att det inte är installerat. I TEX Live och MacTEX kan man installera nya paket med kommandot `tlmgr install` *`paketnamn`*. De flesta paketen kommer även med dokumentation, som kan nås med kommandot `texdoc`, men ibland kan det vara lättare att leta upp motsvarande paket på CTAN och kolla på dokumentationen där istället.

Några paket finns alltid och är mycket viktiga eftersom de berättar för LATEX hur in- och utdata (ska) formateras:

fontenc berättar för LATEX vilken sorts typsnitt det typsatta dokumentet ska använda; de vanligaste är `T3`, `OT1` och `T1` — oftast vill man ha `T1`, som är vektorbaserad:

```
\usepackage[T1]{fontenc}
```

inputenc berättar för LATEX vilken teckenkodning indatan har skrivits i. LATEX (som stammar från 80-talet) antar att teckenkodningen är **ISO-8859-1** om inget annat anges, men många moderna system använder **UTF-8** vilket man då måste berätta:

```
\usepackage[utf8]{inputenc}
```

Från TEX till PDF

För att skapa ett typsatt **PDF**-dokument utifrån en LATEX-fil måste man köra den genom den så kallade kompilatorn. LATEX kommer inte med något fint GUI med behändiga knappar att trycka på[1]; man måste istället köra kompilatorn från terminalen.

Normalt måste man köra kompilatorn åtminstone två gånger, så att LATEX får en chans att förbättra och korrigera småsaker som referenser, innehållsförteckningar med mera[2]. Kompilatorn brukar varna om detta och be om ännu en körning. Använder man vissa externa verktyg som BIBTEX måste man även köra detta program; arbetsflödet blir då LATEX→BIBTEX →LATEX→ LATEX.

Kompilatorn: pdfLATEX

Det finns ett antal olika kompilatorer att tillgå (den vanligaste heter bara `latex` och genererar **DVI**-filer), men den som är att föredra är pdfLATEX. Fördelen med denna kompilator är att den har utökat stöd för vissa mikrotypografiska funktioner, till exempel hängande punktuation (Thành 2000), men framför allt så kompilerar den dina LATEX-filer till **PDF**-dokument, istället för de **DVI**-filer `latex` producerar.

Användningen är enkel; när du vill kompilera ditt dokument (vilket du som sagt vill göra några gånger i rad) anropar du helt enkelt programmet `pdflatex` (testa gärna på koden i exempel 1 på nästa sida):

```
pdflatex filnamn.tex
```

Detta gör att pdfLATEX kompilerar ditt program och skapar ett antal filer, däribland `filnamn.pdf`, som är det slutgiltiga (eller, så slutgiltigt LATEX hunnit göra det) **PDF**-dokumentet.

[1] Även om det finns sådana verktyg, till exempel LyX och TEXnicCenter. [2] Detta är en konsekvens av hur TEX fungerar; kompilatorn läser filen bit för bit och expanderar kommandon — det går inte att röra sig ”bakåt” i filen, så vill man ändra på något man redan gått förbi måste man spara information om detta i en extern fil som man läser in under nästa körning.

```
\documentclass[a4paper,12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\begin{document}
    Det här är ett första stycke, ett exempel
    på hur    text    typsätts    av \LaTeX{}.

    Oj, ett nytt stycke! (Två stycken i samma
    dokument, kan du tänka dig?)
\end{document}
```

Exempel 1: *Ett mycket enkelt LaTeX-dokument med endast det allra nödvändigaste.*

Automatisera med `latexmk`

Det kan vara tröttsamt att konstant kompilera sina filer och hålla koll på hur många gånger man ska köra LaTeX och andra verktyg. Lyckligtvis finns det sätt att automatisera processen, och ett av det enklaste är Perl-skriptet `latexmk`, som finns tillgängligt på CTAN[1]. Skriptet kör LaTeX och tolkar loggen som skapas för att avgöra om det behövs fler körningar, och kan även konfigureras för att kontinuerligt kolla efter ändringar i LaTeX-filen (och andra berörda filer) och kompilera om hela dokumentet när dessa ändras.

Att köra `latexmk` är lika lätt som att köra `pdflatex`:

```
latexmk -pdf filnamn.tex
```

Filer du kanske stöter på

När man arbetar med LaTeX finner man ganska snabbt att det dyker upp en stor mängd olika filer (därför bör man också ha varje dokument i en egen mapp) med mer eller mindre uppenbara filändelser. Vissa behövs för att LaTeX ska fungera, vissa genereras av LaTeX och används i senare körningar och vissa är relativt överflödiga.

Eftersom olika paket också kan skapa filer av godtycklig typ kommer listan nedan inte vara fullständig, men förhoppningsvis innehåller den de filer man oftast stöter på i sitt arbete med LaTeX.

Filer som används av LaTeX

`.tex` Ett TeX- eller LaTeX-dokument. Kompileras med `pdflatex` (eller `latex`, om man vill ha DVI-filer).

`.sty` Ett LaTeX-paket, som kan inkluderas med `usepackage`.

`.dtx` Dokumenterad TeX-kod. Hittar du en sån här fil så är det troligtvis ett distribuerat paket, och det borde följa med en fil med ändelsen `.ins`. Om du kör LaTeX på en `.dtx`-fil så kommer dokumentationen för motsvarande LaTeX-paket genereras.

[1] http://www.ctan.org/tex-archive/support/latexmk/

.ins Installationsfil för en motsvarande .dtx-fil. Kör man LaTeX på denna så kommer den oftast att generera en .sty- eller .cls-fil (detta beror så klart på vad paketet i fråga innehåller).

.cls En dokumentklass. Denna kan väljas med `documentclass`.

Filer som genereras av LaTeX

.pdf PDF-dokument (eng. *Portable Document File*). Det här är sannolikt den fil LaTeX genererat från din .tex-fil, förutsatt att du inte lagrar andra PDF-dokument i samma mapp (vilket du inte bör göra).

.log En detaljerad logg som berättar vad som hände under den senaste körningen av `pdflatex`.

.toc Lagrar alla rubriker, och läses in av LaTeX under nästa körning då den genererar en innehållsförteckning (om en sådan ska finnas med i dokumentet).

.lof Som .toc, fast med en lista över figurer.

.lot Som .toc och .lof, fast med en lista över tabeller.

.aux En fil som innehåller information om korsreferenser, etiketter och annat. Läses in vid nästa körning av `pdflatex`.

II
Typsättning med LaTeX

När du nu vet hur den grundläggande strukturen i ett LaTeX-dokument ser ut är det dags att börja typsätta riktiga dokument. Att typsätta dokument är dock något som tål att tänkas på — den här delen kommer därför inte bara gå igenom vanliga LaTeX-konstruktioner och kommandon, utan även att inleda med en kort diskussion gällande struktur och språk. Efter den här delen bör du med andra ord inte ha några problem med att producera riktiga dokument, varken LaTeX- eller strukturmässigt.

Text- och språkstruktur

Struktur är mycket viktigt för att läsaren ska kunna ta del av de idéer och den information texten förmedlar. Strukturen ska givetvis framgå direkt ur texten, men med väl genomtänkt typografi förstärks denna struktur och hjälper läsaren (och författaren) en bit på vägen.

LaTeX skiljer sig från andra typsättningssystem (och ordbehandlare) på det sättet att du endast behöver berätta för LaTeX vilken logisk och semantisk struktur din text har. Dessutom så främjar det en god struktur genom det sätt språket är uppbyggt. När man berättat för LaTeX vilken struktur texten har tolkar kompilatorn dessa förklaringar efter förbestämda regler (givna av dokumentklassen och de paket som används) och skapar den typografiska strukturen utifrån dessa. Du får alltså vara författaren, och LaTeX tar rollen som typsättare.

Textstycket

Den mest grundläggande byggstenen i LaTeX (och både typografi och författande) är stycket. Vi kallar det ”byggsten” eftersom ett stycke är en typografisk enhet som ska förmedla en tanke eller en idé. Om en ny tanke påbörjas ska alltså även ett nytt stycke påbörjas, och om så inte är fallet ska endast radbrytningar användas[1]. Tvekar du på om du borde ha ett nytt stycke, tänk på din text som en förmedlare av tankar; om du har en styckesindelning men din gamla tanke fortsätter, så ska den troligtvis bort, och om du påbörjar en ny tanke mitt i ett stycke så ska det förmodligen brytas.

Många underskattar vikten av välgenomtänkta styckesindelningar. Andra vet inte ens vad syftet med en styckesindelning är, eller (speciellt i LaTeX) introducerar nya stycken utan att vara medvetna om det. Detta misstag är mycket lätt att göra om till exempel ekvationer används i texten. Ta en snabb titt på exempel 2 på följande sida och försök lista ut varför tomma rader (det vill säga ett nytt stycke) används i vissa tillfällen men inte andra — det är en subtil men mycket viktig skillnad.

[1] Notera dock att det oftast är helt meningslöst att bry sig om radbrytningar i LaTeX, eftersom systemet både avstavar och radbryter bra på egen hand.

```
% Exempel ett
\ldots{}med hjälp av identiteten
\begin{equation}
    \int\limits_{-1}^1\!\sqrt{1-x^2}\;\mathrm{d}x
    = \frac{\pi}{2}
\end{equation}
kan man alltså med Monte Carlo-integration ta fram
ett approximativt värde på \(\pi\).

% Exempel två
\ldots{}Weibullfördelningen, döpt efter den svenska
matematikern Waloddi Weibull, har den kumulativa
fördelningsfunktionen
\begin{equation}
    F(x) = 1 - e^{-(x/\alpha)^\beta} \;.
\end{equation}

Från denna kan vi med derivering\ldots{}

% Exempel tre
\ldots{}relativt ointressanta observationer.

\begin{equation}
    z_{n+1} = z_n^2 + c
\end{equation}
ger å andra sidan upphov till den mycket kända
Mandelbrot-mängden, som\ldots{}
```

Exempel 2: *Tre exempel på korrekt styckesindelning i samband med ekvationer.*

Nästa enhet i sammanhanget är en mening. I engelska texter används ett extra stort mellanrum i slutet av varje mening; detta gör vi inte i svenskan[1] och eftersom LaTeX försöker lista ut var den ska lägga in större mellanrum måste man berätta för LaTeX att den ska låta bli, vilket görs med hjälp av kommandot `\frenchspacing`. Skriver man på engelska ska man alltså utelämna detta kommando, och LaTeX kommer att efter bästa förmåga gissa var dina meningar tar slut. Gör kompilatorn fel får man ersätta det mellanslag den förlänger med ett fast mellaslag (~).

Rubriker

Det andra strukturelementet man måste tänka på[2] är rubriker. Dessa skapar en logisk struktur som delar upp innehållet, och den typografiska effekten av att införa en rubrik är så stark att det nästan är självklart hur man ska använda dessa för att strukturera sin text.

[1] Det finns ganska många saker LaTeX gör som inte bör göras i svensk typografi — det mesta kommer att tas upp senare i den här delen. [2] Undantaget strukturen i meningar, som är den del av svenska språket och som förklaras bättre av Christoffersson et al. (1998).

Standardklasserna i LaTeX har sex rubriknivåer:

```
\part{...}          % Ingen nivå
\section{...}       % Nivå 1
\subsection{...}    % Nivå 2
\subsubsection{...} % Nivå 3
\paragraph{...}     % Nivå 4
\subparagraph{...}  % Nivå 5
```

Dessutom finns det i report- och book-klasserna en rubriknivå till, nivå noll, som ges av `\chapter`. Eftersom article-klassen inte känner igen `\chapter` kan man enkelt inkludera artiklar som kapitel i till exempel en bok.

Rubriknivån som ges av `\part` påverkar inte de andra numreringarna, medan alla andra rubrikkommandon återställer numreringen av lägre nivåer. Detta gör att man kan dela upp sitt dokument i delar, som LaTeX dessutom per automatik numrerar ner till en viss nivå. Denna definieras av dokumentklassen men kan ändras genom att ställa in den inbyggda räknaren `secnumdepth`:

```
\setcounter{secnumdepth}{2} % Numrera ner till subsection
```

Dessutom kan man införa onumrerade rubriker genom att lägga till en stjärna efter rubrikkommandot:

```
\section*{Inledning} % Onumrerad rubrik
```

Dessa rubriker kan sedan om så önskas listas i en innehållsförteckning med hjälp av kommandot `\tableofcontents`, där endast de numrerade rubrikerna inkluderas[1]. Djupet på innehållsförteckningen styrs med räknaren `tocdepth`:

```
\setcounter{tocdepth}{2}
```

Rad- och sidbrytningar

Med LaTeX behöver man i princip aldrig bry sig om att rad- eller sidbryta eftersom detta görs automatiskt. Vid vissa tillfällen kan det dock vara önskvärt att göra det manuellt ändå; framför allt kan det vara intressant att lägga in manuella sidbrytningar för att undvika änkor, horungar eller störande avbrott i textflödet. Tabell 1 på sidan 17 listar några av de kommandon som kan användas för att åstadkomma sidbrytningar.

Notera dock att LaTeX inte lyder kommandoföljder à la ”`\newpage\newpage`” då sidbrytningskommandon endast inforgar en sidbrytning *om det behövs*. Vill man ha en tom sida måste man alltså fylla den med något (osynligt), till exempel en `\mbox` innan man sidbryter en andra gång.

Radbrytningar kan enkelt göras med hjälp av `\\` eller `\newline` i de fall då de behövs; oftast är styckesindelning fullt tillräckligt.

[1] Vill man inkludera även onumrerade rubriker i sin innehållsförteckning måste man uttryckligen lägga till dessa med kommandot `\addcontentsline`. Detta läggs lämpligen till på raden efter rubriken.

Avstavning

Avstavning sköts automatiskt av LaTeX, förutsatt att det finns avstavningsdata för det språk man använder. Denna kan man ladda in med hjälp av paketet babel.

Paketet babel följer med i princip alla LaTeX-installationer. Det innehåller avstavningsregler och översättningar till inbyggda kommandon (rubriker till innehållsförteckning, referenslista och så vidare) för ett stort antal språk, däribland svenska och engelska. Det är enkelt att använda babel (paketet förklaras närmre i del VI), till och med för dokument flera olika språk, och oftast räcker det med att helt enkelt inkludera paketet med rätt inställning:

```
\usepackage[swedish]{babel} % Dokumentet är på svenska
```

Inga avstavningspaket är dock kompletta, och skriver man tekniska rapporter är risken stor att något ord inte finns med eller avstavas på fel sätt. Man kan då berätta för LaTeX hur man vill att ordet ska avstavas med hjälp av kommandot `\hyphenate`. Kommandots argument är en lista av ord (separerade med mellanslag) som har bindestreck placerade där LaTeX får bryta ordet. Sålunda kan raden

```
\hyphenate{FORTRAN span-ku-le-ra}
```

användas för att berätta för LaTeX att ordet ”**FORTRAN**” inte får brytas, medan ordet ”spankulera” får brytas på de angivna positionerna.

Betoning, med mera

För att betona text i exempelvis böcker använder man *kursiv* text. I LaTeX kan man betona text med hjälp av kommandot `\emph`, som i de flesta fall gör just detta; kursiverar texten. Det finns dock andra tillfällen då man använder kursiv text, och det är givetvis dumt att i sådana stycken kursivera för att betona text; i sådana fall använder LaTeX därför helt vanlig, upprätt text (för nästan alla dokumentklasser, i alla fall).

Tabell 2 på nästa sida visar utöver kommandot `\emph` som används för betoning ett antal andra textförändrande kommandon som kan vara bra att känna till. Exempelvis så finns `\textsc`, som typsätter texten med kapitäler (”små versaler”), och `\textsf` som typsätter med ett sans-serif-typsnitt istället för det vanliga serif-typsnittet.

Man bör dock inte använda dessa kommandon direkt, utan istället med hjälp av `\newcommand` definiera kommandon som beskriver *vad* som typsätts, inte *hur* det ska typsättas. Säg till exempel att du vill typsätta namn på LaTeX-paket med sans-serif-typsnitt. Vi kan då definiera ett kommando `\package` som i sin tur tillämpar `\textsf`:

```
\newcommand\package[1]{\textsf{#1}}
```

När detta sedan används i dokumentet ser man direkt att det är ett paket som beskrivs; att det sedan typsätts med sans-serif-typsnitt är inte så intressant just när man skriver dokumentet.

Tabell 1: *Tre kommandon för att skapa sidbrytningar i LaTeX.*

Kommando	Resultat
`\newpage`	Skapar en ny sida.
`\clearpage`	Skapar en ny sida och tvingar alla *floats* (mer om dessa på följande sida) att läggas ut innan den nya sidan. Bättre än `\newpage` om syftet är att bryta för ett nytt kapitel eller liknande.
`\cleardoublepage`	Gör samma sak som `\clearpage`, men ser till att den nya sidan är en udda sida, förutsatt att dokumentet är tvåsidigt. Gör exakt samma sak som `\clearpage` i enkelsidiga dokument.

Tabell 2: *Viktiga LaTeX-kommandon för att ändra textstil.*

LaTeX-kommando	Resultat	Kommentar
`\textnormal{…}`	Dokumentets standardtypsnitt	Det typsnitt som används för ”vanlig” text i dokumentet.
`\emph{…}`	*Betonad text*	Ska alltid användas för betoning då kommandot automatiskt hanterar nästlade betoningar på ett bra sätt.
`\textrm{…}`	Serif-typsnitt	
`\textsf{…}`	Sans serif-typsnitt	
`\texttt{…}`	`Teletype-typsnitt`	Detta typsnitt har fast bredd och kan användas för att typsätta till exempel kodstycken.
`\textit{…}`	*Kursiv text*	Använd `\emph` istället för detta kommando för att betona text.
`\textbf{…}`	**Fetstilt text**	Använd `\emph` istället för detta kommando för att betona text.
`\textsc{…}`	KAPITÄLER	

Figurer, tabeller, listor

Figurer, tabeller och listor[1] har i sin LaTeX-representation en sak gemensamt; de kräver mer än enkla kommandon för att kunna uttryckas på ett enkelt sätt. Därför har man i LaTeX valt att införa en konstruktion som grupperar sådana företeelser och gör det tydligt hur textens struktur verkligen ser ut — denna konstruktion kan på svenska kallas för en omgivning.

Omgivningar (eng. *environments*)

Omgivningar i LaTeX definieras utifrån två kommandon; `\begin` och `\end`. De kan nästlas (på det sättet att en omgivning kan innehålla en annan) men kan inte överlappa — varje omgivning måste avslutas innan dess ”förälder” avslutas, vilket gör att de ger en tydlig trädliknande struktur. En omgivning (här `verbatim`) kan alltså skrivas

```
\begin{verbatim}
    Det här är omgivningens innehåll
\end{verbatim}
```

där omgivningens innehåll (oftast) med fördel kan indenteras för att göra strukturen tydlig.

Det finns många omgivningar, och de fyller alla olika syften. De viktigaste bildar listor, figurer, tabeller och ekvationer, men en omgivning kan i princip göra vad som helst.

Tre sorters listor

De första omgivningarna vi ska bekanta oss med är de som skapar listor. I LaTeX finns det tre användbara sorters listor, som alla går att nästla i varandra utan problem och som löser tre distinkta problem: numrerade listor (`enumerate`), onumrerade listor (`itemize`) och beskrivningslistor (`description`). Dessa används tillsammans med kommandot `\item` för att bygga enkla listor (se exempel 3 på motstående sida) och är relativt självförklarande.

Flytande objekt (eng. *floats*)

Tabeller och figurer är viktiga verktyg för att förmedla resultat på ett effektiv sätt. De är också mycket speciella objekt rent typografiskt, eftersom de till skillnad från vanliga textstycken inte kan brytas över sidor hur som helst. För att placera figurer och tabeller på ett tillfredsställande sätt använder LaTeX sig av så kallade *floats*, vilket innebär att objektet får ”flyta iväg” en bit ifrån det ställe där det från början definieras. Detta undviker fula halvtomma sidor och andra obehagliga resultat.

I många böcker typsätts figurer och tabeller så att de antingen flyter till toppen av en sida, botten av en sida, eller så att de tar upp en hel sida. Man typsätter i princip aldrig figurer och tabeller så att de bryter texten. Man kan även göra samma sak med till exempel kod, men detta är inte lika vanligt. Även om det finns sätt att tvinga LaTeX att lägga en figur där den definieras så rekommenderas detta alltså inte.

Ett flytande objekt definieras av en omgivning, och har ett valfritt argument som berättar för LaTeX hur objektet får placeras. Detta kan innehålla bland annat `t` (*top*), `p` (*page*) och `b` (*bottom*)

[1] Men även ekvationer, programkod, vissa kemiska formler, citat och algoritmer, för att nämna några ytterligare exempel.

```
\begin{enumerate}
  \item Skapa en lista
  \item Lägg till ett element
  \item Upprepa steg 2 om så
        önskas
\end{enumerate}
```

1. Skapa en lista
2. Lägg till ett element
3. Upprepa steg 2 om så önskas

(**a**) *En numrerad lista*

```
\begin{itemize}
  \item Tyskland
  \item Topologi
  \item Teknolog
\end{itemize}
```

- Tyskland
- Topologi
- Teknolog

(**b**) *En onumrerad lista*

```
\begin{description}
  \item[Häst] ett stort,
        fyrbent däggdjur.
  \item[Öl] en mycket god
        dryck.
\end{description}
```

Häst ett stort, fyrbent däggdjur.
Öl en mycket god dryck.

(**c**) *En beskrivningslista*

Exempel 3: *De tre enkla listomgivningar LaTeX tillhandahåller.*

och är en lista över alla de sätt LaTeX tillåts placera objektet. Om inget av dessa fungerar kommer LaTeX att placera objektet på en egen sida. Sålunda kan man skapa en *float* med följande kod:

```
\begin{<floattyp>}[tbp] % Får placeras som top eller bottom
    % Innehåll i det flytande objektet
\end{<floattyp>}
```

Flytande objekt kan även ha figurtexter, som definieras med kommandot `\caption`. En figurtext ska komplettera figuren och förklara vad som visas, inte varför det visas. Slutsatser och liknande dras istället i brödtexten som refererar till figuren. Figurer har sin figurtext undertill medan tabeller har figurtexten ovanför tabellen.

Figurer

Figurer är den kanske vanligaste typen av flytande objekt, och dessa (grafer, illustrationer, diagram och så vidare) infogas i omgivningen `figure`, vilken visas i exempel 4 på nästa sida.

Att infoga själva figuren kan göras på några olika sätt; man kan skapa den med ett externt program (till exempel `gnuplot` eller Inkscape), exportera den direkt från MATLAB eller Mathematica, eller till och med rita den direkt i LaTeX. Hur man gör detta diskuteras närmre i del IV på sidorna 45–46.

För att centrera innehållet i figuren använder man kommandot `\centering`, som centrerar allt innehåll i resten av omgivningen. En del får för sig att istället använda omgivningen `center`, men eftersom den omgivningen lägger till extra tomrum både före och efter är det lämpligare att bara använda `\centering`.

Tabeller

Tabeller är ett annat vanligt flytande objekt, och är ett alldeles förträffligt verktyg när det gäller presentation av data som till exempel mätserier eller liknande. LaTeX definierar två tabellrelaterade omgivningar; `table` och `tabular`. Det förstnämnda är ett flytande objekt och används precis som `figure`:

```
\begin{table}[tpb]
    \centering
    \caption{En beskrivning av tabellen}
    % Här ska själva tabellen in
\end{table}
```

Den andra omgivningen, `tabular`, är den som faktiskt används för att definiera tabellen. LaTeX kan i denna omgivning hjälpa till med att rita linjer både kors och tvärs, och definiera kolumner på ett antal olika sätt, men för att hålla sig till en enkel (om än bokinspirerad) devis så bör man ta paketet booktabs till hjälp när man skapar tabeller i LaTeX[1]. En enkel tabell skapad med hjälp av booktabs kan se ut som tabellen i exempel 5 på motstående sida.

Som vi ser tar `tabular`-omgivningen ett obligatoriskt argument, som beskriver de kolumner tabellen innehåller. Tabell 3 på sidan 22 listar de vanligaste kolumntyperna och deras användningsområden. Kolumntyperna anges direkt efter varandra, men kan om man vill separeras med

[1] Exakt varför detta är en bra idé förklaras mycket bra av booktabs-manualen (Fear 2005).

```
\begin{figure}[tbp]
  \centering
  \framebox[0.75\textwidth]{
  \medskip PLACEHOLDER \medskip
  }
\caption{En kort figurtext}
\end{figure}
```

Figure 1: En kort figurtext

Exempel 4: *Ett exempel på hur man skapar ett flytande objekt med* `figure`.

```
\begin{tabular}{l r p{4cm}}
    \toprule
    Konstant & Värde & Kort beskrivning \\
    \midrule
    \(\gamma\) & \(0,577\) & Skillnaden mellan den
     harmoniska summan från \(1\) till \(n\) och den
     naturliga logaritmen av \(n\) då \(n\to\infty\).\\
    \(e\) & \(2,718\) & Den konstant som har
    egenskapen att \((e^x)' = e^x\). \\
    \(\pi\) & \(3,1415\) & Kvoten mellan en cirkels
     omkrets och dess diameter. Mycket viktig
     konstant i många sammanhang; dyker upp
     lite varstans. \\
    \bottomrule
\end{tabular}
```

Konstant	Värde	Kort beskrivning
γ	0,577	Skillnaden mellan den harmoniska summan från 1 till n och den naturliga logaritmen av n då $n \to \infty$.
e	2,718	Den konstant som har egenskapen att $(e^x)' = e^x$.
π	3,1415	Kvoten mellan en cirkels omkrets och dess diameter. Mycket viktig konstant i många sammanhang; dyker upp lite varstans.

Exempel 5: *En tabell skapad med hjälp av* `booktabs`.

Tabell 3: *De fyra viktigaste kolumntyperna i* `tabular`.

Typ	Kommentar
`l`	Vänsterjusterar innehållet. Använd till text, datum och så vidare.
`c`	Centrerar innehållet. Använd inte.
`r`	Högerjusterar innehållet. Använd till tal, mätresultat och så vidare.
`p{...}`	Avstavat textstycke med angiven bredd. Använd när texten är så lång att tabellen bli för bred för sidan, eller när du skriver långa beskrivningar och/eller kommentarer.

exempelvis mellanslag. Den speciella separatorn `|` skapar en vertikal linje mellan två kolumner, men detta är inte att rekommendera (Fear 2005).

Efter att omgivningen inletts med `\begin` följer tabellens rader. Här måste varje rad markeras med explicita nyrader (\\), och kolumnerna separeras med och-tecken (&). Notera även de kommandon som skapar horisontella linjer; dessa är från booktabs och används enligt exempel 5.

Det är även viktigt att komma ihåg några typografiska regler när det gäller tabeller. Alla dessa ges av Fear (2005, s. 3). Fritt översatt:

1. *Använd aldrig någonsin lodräta linjer.*
2. *Använd aldrig dubbla linjer.*
3. *Lägg enheter i tabellhuvudet.*
4. *Decimalkomma ska alltid föregås av en siffra; alltså* 0,1 ***inte*** *bara* ,1.
5. *Använd inte upprepningstecken eller liknande konventioner för att repetera föregående värde. I många fall räcker det med en tom cell. Gör det inte det, repetera värdet.*

Punkt ett och två är enkla att följa om man använder booktabs som manualen föreskriver, och punkt tre och fem är snarare arbetsbesparande än ansträngande att hålla sig till. Även punkt fyra är enkel att följa. Följer man dessa fem punkter får man snygga tabeller varje gång[1].

Etiketter och korsreferenser

Eftersom figurer får (och bör få) flyta en bit bort, i vissa fall till andra sidor, måste man kunna referera till dem i texten. Detta görs med så kallade korsreferenser till etiketter. En etikett kan man definiera med kommandot `\label`, inte bara för figurer och tabeller utan även för rubriker, ekvationer och annat. För flytande objekt och andra omgivningar placerar man `\label`-kommandot inuti omgivningen (men efter `\caption` om denna används), medan man för rubriker placerar etiketten direkt efter `\section` (eller motsvarande).

[1] Läs gärna igenom Fear (2005) ordentligt de tre-fyra första gångerna du sitter med tabeller i LaTeX, så går användandet in i ryggmärgen.

En etikett kan innehålla många olika tecken, men det kan vara bra att hålla sig till (amerikanska) bokstäver, siffror samt kolon (:) som separator. För att kunna hålla koll på sina etiketter bör man även namnge dem på ett logiskt sätt (varje etikett bör beskriva det den refererar till) och gärna inleda varje etikett med en liten beskrivande förkortning. Man kan till exempel märka en figur som visar β-sönderfall med följande etikett:

```
\label{fig:betasonderfall}
```

Etiketterna kan man sedan referera till med `\ref`, som skriver ut det nummer figuren (eller motsvarande) har. Dessutom finns `\pageref`, som skriver ut vilken sida figuren ligger på. Dessa kommandon är dock inte medvetna om vilken typ av objekt man refererar till, så detta måste man berätta:

```
...som man ser i figur~\ref{fig:betasonderfall}...
```

Paketet varioref inför det något mer intelligenta kommandot `\vref` (och även `\vpageref`) som formulerar referensen på ett något bättre sätt, beroende på var figuren ligger. Ligger figuren på samma sida är `\vref` i princip ekvivalent med `\ref`, men om figuren är på en annan sida skriver `\vref` även ut vilken sida detta är.

Paketet hyperref förbättrar både `\ref` och `\vref` genom att göra referenserna till länkar i den slutgiltiga PDF-filen. Med dessa kan man alltså lättare navigera i dokumentet. Förutom detta så lägger hyperref dessutom till en innehållsförteckning som PDF-läsaren kan visa i programmet, och annan PDF-specifik funktionalitet. Paketet rekommenderas starkt om du använder pdfLaTeX.

Typografiska betänkligheter

När man skriver rapporter är det även viktigt att hålla sig till de typografiska regler LaTeX inte håller reda på; sådana som har med citattecken, tankstreck och annat att göra. Även datum, decimalavskiljare och liknande är viktigt att tänka på. Dessa typografiska regler kommer i korthet att förklaras på de närmsta sidorna, till vilka viss inspiration tagits från von Schultz (2005).

Citattecken

Det mest grundläggande man bör uppmärksamma i LaTeX, eftersom det är en grop man lätt faller i, är citattecken. Man kan inte i LaTeX använda det vanliga citattecknet (") för att generera ett citattecken, utan man måste istället använda kombinationer av *backticks* (`) och apostrofer ('). Tabell 4 på följande sida visar hur dessa används för att citera text.

Streck av olika längd

Det kan tyckas vara ett rent typografinörderi man inte behöver bry sig om, men faktum är att det finns olika sorters streck (lite på samma sätt som att det finns olika skiljetecken). Det är relativt viktigt att skilja på dessa och se till att använda rätt streck vid rätt tillfälle.

Det första strecket, minustecknet, är relativt långt (−) och går i princip inte att missa eller missbruka. Det infogas av LaTeXdå man använder ett vanligt bindestreck (-) i matematikläge[1]:

```
...och ett flyttal kan därför ha värdet \(-0\).
```

[1] Mer om detta i del III.

Tabell 4: *Citattecken i LaTeX.*

LaTeX-kod	Resultat	Kommentar
``Text''	“Text”	Engelska citattecken. Använd inte i svensk text!
`Text'	‘Text’	Nästlade engelska citattecken (för citat inuti citat). Använd inte i svensk text!
''Text''	”Text”	Svenska citattecken.
'Text'	’Text’	Nästlade svenska citattecken (för citat inuti citat).

Nästa tecken, det vanliga bindestrecket (-), behöver man i princip aldrig använda själv. Det infogas av LaTeX då ord avstavas, men kan även infogas manuellt genom att helt enkelt skriva ett vanligt bindestreck i LaTeX-koden.

Det tredje strecket kallas intervallstreck (–) och är något längre än bindestrecket. Detta används som namnet antyder för att typsätta intervall, och skrivs med hjälp av två bindestreck (--):

```
...det kan ta upp emot 3--4 timmar att...
```

Det fjärde och sista strecket kallas långt tankstreck (—) och är det längsta av strecken. Det långa tankstrecket är relativt ovanligt inom svensk typografi, men det finns ingen anledning att undvika det[1]. Ett långt tankstreck skriver man med tre bindestreck i följd (---), och man bör omge det med mellanslag:

```
...det är --- för tillfället åtminstone --- omöjligt att...
```

Avstånd mellan stycken

Det finns två vedertagna sätt att separera stycken; indrag och mellanrum. Man tillämpar endast *ett* av dessa åt gången, och traditionellt sett så brukar man i svensk typografi använda mellanrum, medan man i amerikansk typografi använder indrag (varför detta är vad LaTeX gör *out-of-the-box*).

För att få LaTeX att använda mellanrum istället för indrag bör man egentligen skriva helt nya dokumentklasser, men eftersom detta är ett stort jobb kan man istället ”(eng. *patcha*)” LaTeX. Detta görs av paketet parskip:

```
\usepackage{parskip}
```

[1] För tankar (*no pun intended*) angående användningen av tankstreck, se Christoffersson et al. (1998, ss. 46–47) — observera dock att man *inte* som Christoffersson et al. föreslår ska använda tankstrecket i intervall eller punktlistor. Det Språkrådet (2008) och Christoffersson et al. betecknar som ”tankstreck” är i själva verket mycket närmre det som i LaTeX anses vara ett intervallstreck, det vill säga ett lite kortare tecken. Personligen föredrar jag att göra skillnad på de två, och gör man det bör intervallstrecket användas till intervall, och det långa tankstrecket som alternativ till kommatecknet.

Andra saker att hålla reda på

- Datum skrivs på bästa sätt i ISO-format (YYYY–MM–DD), där man använder intervallstreck för att skilja delarna åt.
- Decimalavskiljare i svensk skrift är kommatecknet. För att LaTeX inte ska lägga in mellanrum efter kommatecken i matteläge (LaTeX tolkar det som en koordinat) bör man inkludera paketet icomma. I engelsk skrift används punkt som decimalavskiljare.
- Enheter ska typsättas med ett (litet) mellanrum mellan tal och enhet. Detta görs enklast med paketet siunitx, som förklaras närmre på sidan 44.

Skarpare typsnitt: fontenc och lmodern

Eftersom LaTeX och dess tillhörande typsnitt (Computer Modern) är relativt gamla, och eftersom de designats för att fungera väl med det mycket gamla PostScript-formatet, så uppstår det ibland några smärre problem. Det första problemet som brukar dyka upp i samband med typsnitt i LaTeX är att texten blir otydlig och suddig i det typsatta PDF-dokumentet. Detta beror på att de typsnitt som används är baserade på bitmappar, som inte skalar särskilt väl. Därför måste man använda paketet fontenc för att säga åt LaTeX att använda vektorbaserade typsnitt istället:

```
\usepackage[T1]{fontenc}
```

Det är då man brukar stöta på nästa problem, som yttrar sig i att text (främst skandinaviska tecken och andra tecken som inte finns representerade i ANSI) inte kopieras på ett korrekt sätt. Detta är en konsekvens av att de mycket gamla Computer Modern-typsnitten inte innehåller de glyfer som motsvarar sådana tecken, och istället används då (något fulhackade) komposita tecken istället. För att lösa detta måste man byta typsnitt.

Det typsnitt som brukar användas istället för Computer Modern är dess mer moderna klon *Latin Modern*. Detta typsnitt ser i princip ut som Computer Modern, men innehåller alla de glyfer som behövs för att typsätta svensk (och annan icke-anglikansk) text. För att använda Latin Modern inkluderar man paketet lmodern:

```
\usepackage{lmodern}
```

Andra viktiga delar

Även om innehållet är det viktigaste så finns det andra delar av en rapport eller artikel som också måste ges lite uppmärksamhet. Titelsidor bör man till exempel alltid ha, och det är även viktigt att ha koll på hur man citerar och vilken teckenkodning man använder. I vissa fall kan man även vilja referera till vissa längder, eller använda kommandon som LaTeX anser vara ”ömtåliga” i situationer de inte är tänkta att användas. Avslutningen på detta kapitel kommer därför att gå igenom dessa småsaker.

Titelsidan

Det kan vara trevligt att i sitt arbete ha en fin titelsida. Även denna kan LaTeX (åtminstone standardklasserna) typsätta åt dig. Detta är enkelt att göra, och fungerar i princip så att man definierar sin metadata (titel, datum, författare) med hjälp av ett par kommandon, varpå man kör kommandot `\maketitle` för att skriva ut en titelsida:

```
\title{En fantastisk rapport\thanks{Tack till DD-gudarna
       för visad barmhärtighet och stabila filservrar.}}
\date{2011--12--13}
\author{E.~Johansson\thanks{erik.johansson@example.org}}
\maketitle
```

Kommandot `\thanks` kan användas för att infoga korta tack eller extra information om titel eller författare, så som mailadresser eller liknande. Den typografiska effekten av `\thanks` (i standardklasserna) än en fotnot.

Man kan dessutom infoga en sammanfattning (eng. *abstract*) med hjälp av omgivningen `abstract`. Denna omgivning inkluderas lämpligen direkt efter `\maketitle`, och skapar automatisk ett litet textstycke med rubriken ”Sammanfattning” (eller, om engelska valts som språk, ”Abstract”).

Vill man (säg, i ett kandidatarbete) ha både *abstract* och sammanfattning måste man använda babels funktioner för att byta språk mitt i dokumentet. Man måste då först ladda in babel med stöd för båda språken (notera att det sista språket i listan är aktivt från dokumentets start):

```
\usepackage[english,swedish]{babel}
```

Därefter byter man helt enkelt språk när man vill ha sitt *abstract*:

```
\begin{otherlanguage}{english}
    \begin{abstract}
        % Abstract
    \end{abstract}
\end{otherlanguage}
\begin{abstract}
    % Sammanfattning
\end{abstract}
```

Citat

Det finns två omgivningar i LaTeX som kan användas för att märka upp så kallade blockcitat (eng. *blockquotes*), alltså lite längre citat på ett eller flera stycken; `quote` och `quotation`. Det första, `quote`, är tänkt att användas till citat på högst ett stycke, medan `quotation` ska användas för flera stycken.

Teckenkodning (UTF-8)

Teckenkodning, som kort nämndes på sidan på sidan 9, är viktigt att hålla koll på. Det finns två större teckenkodningar på Linux-plattformar, UTF-8 och ISO-8859-1, och en till på Windows-

system, CP-1252. Det är viktigt att man vet vilken teckenkodning man skriver sina dokument i, då man måste berätta detta för LaTeX. Gör man inte det förutsätter LaTeX att dokumentet är skrivet i ISO-8859-1, och är det då inte det så kommer hemska saker hända. Teckenkodningen specificeras med paketet inputenc:

```
\usepackage[<teckenkodning>]{inputenc}
```

Här refererar man till UTF-8 som `utf8`, ISO-8859-1 som `latin1` och CP-1252 som `ansinew`. Det finns även en del mer obskyra teckenkodningar[1], men det är troligtvis de tre nämnda du kommer att stöta på.

Längder

Titt som tätt behöver man i LaTeX använda längder för att till exempel kontrollera bredden på figurer och liknande. Den viktigaste fördefinierade längden är `\textwidth`, som representerar bredden av det område på sidan som får fyllas med text. Genom att lägga till ett flyttal innan längden kan man dessutom berätta för LaTeX att man till exempel vill ha en figur som endast täcker 3/4 av bredden:

```
0.75\textwidth
```

Det finns en stor mängd längder i LaTeX, och många av dem relaterar till olika mått på sidan (Oetiker et al. 2011, s. 132), medan andra (till exempel `\smallskipamount`) är mer obskyra och godtyckliga.

Ömtåliga kommandon

En del text, så som rubriker, figurtexter med mera, kan dyka upp mer än en gång i ett dokument (till exempel i innehållsförteckningen). Argument till sådana kommandon kallas för rörliga (eng. *moving arguments*) och kan ställa till med en del problem. Det finns nämligen kommandon (`\footnote`, till exempel) som är ”ömtåliga” (eng. *fragile*), och som därför inte kan användas i rörliga argument utan vidare. Vill man göra det måste man skydda dem, något som görs med kommandot `\protect`:

```
\section{En rubrik\protect\footnote{Med en fotnot.}}
```

[1] De som stöds av inputenc listas i paketets manual.

III
Matematik med LaTeX och AMS

LaTeX är ett mycket bra system för typsättning, och ett av de områden LaTeX är starkast inom är typsättning av matematik. För att utöka och förenkla de grundläggande mekanismer LaTeX har för typsättning av matematik har *American Mathematical Society*, AMS, skapat ett antal paket (främst amsmath och amssymb) som kollektivt refereras till som AMS-LaTeX och som gör typsättning av matematik med LaTeX mycket bättre.

Dessa paket är dock enorma, och det finns hela böcker tillägnade endast typsättning av matematik i LaTeX (till exempel Voß 2010, som för övrigt är en mycket bra referens att ha till hands när man typsätter matematik i LaTeX). Den här delen kommer att ge en kort introduktion till de allra enklaste och vanligaste konstruktionerna man stöter på när man typsätter matematik i LaTeX, och tar mycket inspiration från Voß (2010).

För att typsätta matematik så som det beskrivs i den här delen måste du använda paketen amsmath och amssymb, som bör följa med din LaTeX-distribution:

```
\usepackage{amsmath}
\usepackage{amssymb}
```

Att visa ekvationer

Ekvationer är mångfacetterade och man kan vilja inkludera ekvationer av olika anledningar och på olika sätt. I LaTeX kan figurer infogas *inline* (det vill säga i löpande text) eller i omgivningar (vilket frilägger ekvationen, något som är lämpligt för längre ekvationer eller ekvationer av större vikt).

Ekvationer i löpande text

Ekvationer i löpande text kan infogas på två olika sätt; dels genom att placera matematiken mellan dollartecken (`$\sin(x)$`) och dels genom att använda de LaTeX-specifika alternativen till dollartecken, `\(` och `\)`. Det rekommenderas att använda det senare, eftersom detta inte ger fullt så obskyra felmeddelanden när saker är fel, men även dollartecknen fungerar bra och det är ingen typografisk skillnad.

Matematik som typsätts i löpande text kommer att anpassas i höjd för att inte störa texten omkring. Det innebär att till exempel kvoter kommer att se tilltryckta ut. Oftast bör man dock inte använda matematik i löpande text vid de tillfällen då det kommer se ”fel” ut, eftersom ett block då är mer logiskt, textmässigt. Matematik i löpande text är istället oftast konstanter, enstaka värden och kanske någon olikhet.

Ekvationer som ekvationer

Längre och viktigare ekvationer bör inte typsättas i löpande text — man bör istället använda en av de många omgivningar (AMS)LaTeX ger tillgång till. Den allra enklaste omgivningen (som fungerar i de flesta fall) är `equation`, som finns i en vanlig variant som skapar en numrerad ekvation samt en stjärnvariant (`equation*`) som ger en onumrerad ekvation. Som en tumregel bör man endast använda den numrerade varianten om man behöver referera till ekvationen, eller om det är en viktig definition som andra kan tänkas vilja referera till.

Fristående ekvationer kan typsättas mycket bättre eftersom de inte är begränsade av omgivande text. Integraler och summor kan göras så höga som det krävs, och paranteser kan växa nästan obegränsat. Det är dock viktigt att komma ihåg att en fristående ekvation *oftast* inte ska vara ett eget stycke. Exempel 2 på sidan 14 visade hur ett antal ekvationer och mellanrumet mellan dessa och den omgivande texten varierade beroende på ekvationens plats i stycket. Trots att ekvationen är ”fristående” bör man alltså behandla den precis som en ekvation i löpande text. Skillnaden är främst typografisk.

Referera till ekvationer

Likt figurer och tabeller finns det även ett behov av att kunna referera till sina ekvationer. På samma sätt som för figurer markerar man den ekvation man vill referera till med hjälp av `\label`, men istället för att referera med hjälp av `\ref` eller `\vref` bör man använda `\eqref` för att referera till ekvationer. Sådana referenser typsätts lite annorlunda, och bör inte heller föregås av någon text som indikerar att det är en referens.

En snabbkurs i LaTeX-matematik

Nu när du vet hur man skapar förutsättningarna för att typsätta matematik är det dags att gå in på detaljerna. Det finns många konventioner och betänkligheter när det gäller typsättning av matematik, så det är svårt att göra något annat än att skrapa på ytan, men sidorna framöver ämnar presentera tillräckligt mycket för att du ska klara att typsätta det mesta du stöter på under en typisk kandidatutbildning i Teknisk Fysik eller Teknisk Matematik.

Till att börja med måste två elementära operationer förklaras; *superskript* (för potenser och dylikt) och *subskript* (för index och liknande). Ett superskript inför man i LaTeX med hjälp av en cirkumflex (`^`); koden `\(e^x\)` producerar alltså e^x. Ett subskript å andra sidan införs med hjälp av ett understreck (`_`) på liknande sätt; `\(x_i\)` blir x_i. Notera att dessa tecken endast påverkar nästa *kommandoföljd*, vilket innebär att man om man vill upphöja/nedsänka mer än ett tecken (eller kommando), så måste man omsluta dessa med måsvingar:

```
\(x_{i,j}\)
```

Detta innebär även att om man vill ha både superskript och subskript efter en variabel så är det inget problem; man inkluderar bara dessa i följd direkt efter variabeln (eller operatorn):

```
\(x_i^2\)
```

Dessa två exempel producerar alltså $x_{i,j}$ och x_i^2, respektive. Det skall även nämnas att man kan sätta sub- och superskript framför en tom grupp, vilket är användbart om man vill beteckna

Tabell 5: *Enkla matematiska konstruktioner i LaTeX.*

Konstruktion	LaTeX-kod	Resultat
Kvadratrot	`\sqrt{x}`	$\sqrt{x}$
n-rot	`\sqrt[n]{x}`	$\sqrt[n]{x}$
Binomialkoefficient	`\binom{n}{k}`	$\binom{n}{k}$
Kvot	`\frac{a}{b}`	$\frac{a}{b}$

exempelvis isotoper: `{}_{92}^{235}\textrm{U}` kan till exempel beteckna en uranatom och typsätts ${}_{92}^{235}\mathrm{U}$.

Det finns även några viktiga skillnader mellan matematiktypsättning och vanlig texttypsättning, som främst har att göra med mellanrum:

- Mellanslag i koden ignoreras nästan alltid. De mellanrum som ska visas i det typsatta dokumentet placeras istället ut av LaTeX. Det finns även explicita kommandon som gör detta. Detta gör att du kan vara liberal med mellanrum i din kod, vilket ökar läsbarheten.
- Tomma rader tillåts inte i matematiktypsättning. Endast ett ”stycke” per ekvationsomgivning får förekomma.
- Alla bokstäver anses vara variabler och typsätts därför med kursiv text. De bokstäver som inte är variabler (till exempel differentieringsoperatorn d) måste skrivas in med kommandot `\mathrm{...}`. Längre bitar text (eller snarare text som egentligen skulle kunna vara brödtext) bör skrivas in med kommandot `\text`.

Enkla konstruktioner

Det finns ett antal enkla konstruktioner inom matematiken som var man (och kvinna) bör känna till och kunna typsätta. Dessa är konstruktioner som kvoter, binomialkoefficienter, rötter och så vidare. Dessa visas kollektivt i tabell 5.

Ibland kan det även vara önskvärt att markera bitar av ekvationer med små kommentarer ovan och under matematiken. Detta kan göras med kommandona `\overbrace` och `\underbrace`, vilket visas i exempel 6.

```
\begin{equation*}
  \underbrace{y'' + 2y' + y}_{\mathrm{D}(y)}
  = \overbrace{ax^2+bx+c}^{f(x)}
\end{equation*}
```

$$\underbrace{y'' + 2y' + y}_{\mathrm{D}(y)} = \overbrace{ax^2 + bx + c}^{f(x)}$$

Exempel 6: *Annotering med `\overbrace` och `\underbrace`.*

Tabell 6: *Diverse användbara symboler i LaTeX.*

Kommando		Kommentar
`\hbar`	$\hbar$	Konstant inom kvantfysiken
`\Re`	$\Re$	Realdel av komplext tal
`\Im`	$\Im$	Imaginärdel av komplext tal
`\aleph`	$\aleph$	Beskriver kardinalitet av mängder
`\forall`	$\forall$	
`\exists`	$\exists$	
`\nexists`	$\nexists$	
`\complement`	$\complement$	
`\emptyset`	$\emptyset$	Tomma mängden; även `\varnothing` ($\varnothing$) eller `\mathcal{O}` ($\mathcal{O}$)
`\infty`	∞	Oändlighet

Funktioner etc.

Många välanvända eller kända funktioner och operatorer (trigonometriska funktioner och liknande) har långa namn som bör skrivas i upprätt text (det vill säga med `\mathrm`), men då detta är både tidskrävande och inelegant ur en semantisk synpunkt definierar (AMS)LaTeX några av dessa som kommandon istället — sinus blir `\sin`, cosinus `\cos` och så vidare.

De vanligaste trigonometriska funktionerna, logaritmer samt vanliga operationer så som max, min, sup, inf och lim finns som kommandon, och om man behöver typsätta en sådan funktion kan man oftast satsa på att den finns. Finns den inte så kan man skapa den med hjälp av kommandot `\DeclareMathOperator`:

```
\DeclareMathOperator{gcd} % Största gemensamma nämnare
```

Symboler

Matematiken innehåller ett stort antal symboler, däribland det grekiska alfabetet, talmängder, oändligheter, tomma mängden och så vidare (och då inkluderar vi inte de pilar, operatorer, hattar och olikhetstecken som diskuteras senare). Några av dessa symboler listas i tabell 6 till 7 på sidorna 32–33 — notera att några av de grekiska bokstäverna inte har några kommandon för versaler, eftersom dessa är identiska med latinska bokstäver, samt att några har ”varianter” som ser något annorlunda ut (till exempel `\varepsilon`, som är mer lik det lilla epsilon man brukar rita för hand än `\epsilon`). Notera även att de inte typsätts i kursiv stil, vilket oftast är lämpligt. I så fall bör man använda paketet fixmath, som ser till att även de grekiska bokstäverna typsätts som de variabler de faktiskt brukar vara.

Det är givetvis svårt (en del skulle säga omöjligt) att hålla reda på motsvarande LaTeX-kommandon för alla dessa olika symboler själv, och det kan därför vara bra att ha en bra referens (till exempel Pakin 2009). Man kan även använda det utmärkta verktyget Detexify[1], i vilket man kan rita sin symbol och få en lista över liknande LaTeX-symboler och deras motsvarande kommandon.

[1] http://detexify.kirelabs.org/classify.html

Tabell 7: *Kommandon motsvarande symbolerna i grekiska alfabetet.*

Gemen		Versal		Variant	
\alpha	α	\mathrm{A}	A		
\beta	β	\mathrm{B}	B		
\gamma	γ	\Gamma	Γ		
\delta	δ	\Delta	Δ		
\epsilon	ϵ	\mathrm{E}	E	\varepsilon	ε
\zeta	ζ	\mathrm{Z}	Z		
\eta	η	\mathrm{H}	H		
\theta	θ	\Theta	Θ	\vartheta	ϑ
\iota	ι	\mathrm{I}	I		
\kappa	κ	\mathrm{K}	K		
\lambda	λ	\Lambda	Λ		
\mu	μ	\mathrm{M}	M		
\nu	ν	\mathrm{N}	N		
\xi	ξ	\Xi	Ξ		
o	o	\mathrm{O}	O		
\pi	π	\Pi	Π	\varpi	ϖ
\rho	ρ	\mathrm{P}	P	\varrho	ϱ
\sigma	σ	\Sigma	Σ	\varsigma	ς
\tau	τ	\mathrm{T}	T		
\upsilon	υ	\Upsilon	Υ		
\phi	ϕ	\Phi	Φ	\varphi	φ
\chi	χ	\mathrm{X}	X		
\psi	ψ	\Psi	Ψ		
\omega	ω	\Omega	Ω		

Tabell 8: *Hattar och dylikt med LaTeX-kommandon.*

Notation		Kommentar
`\hat{a}`	$\hat{a}$	Transformerade variabler
`\vec{a}`	$\vec{a}$	Vektorer (alternativ notation existerar)
`\dot{a}`	$\dot{a}$	Förstaderviata med avseende på tid
`\ddot{a}`	$\ddot{a}$	Andraderivata med avseende på tid
`\a'`	a'	Generell förstaderivata
`\a''`	a''	Generell andraderivata
`\bar{a}`	$\bar{a}$	Statistiskt medelvärde
`\tilde{a}`	$\tilde{a}$	Statistisk median
`\widehat{abc}`	$\widehat{abc}$	Variant av `\hat` för flera tecken

Operatorer

Förutom symbolerna som nyss diskuterades finns även en mycket stor mängd operatorer inom matematiken. Även dessa kan man leta fram med hjälp av Detexify, men de flesta (åtminstone de som används frekvent) har logiska namn; nabla-operatorn (∇) heter till exempel `\nabla`, och tillhörandeoperatorn ($\in$) heter `\in`. Listor över de vanligaste pilarna, relationssymbolerna och operatorerna i LaTeX återfinns i tabell på motstående sida.

Hattar, prickar, primtecken och liknande

Ibland vill man även modifiera variablers betydelse, kanske för att beteckna en transformerad variabel, en vektor eller en derivata med avseende på tid. Detta kan göras med de LaTeX-kommandon som listas i tabell 8, tillsammans med en kort förklaring av notationens innebörd.

Just för vektorer existerar det dock flera olika notationer. En populär variant (som LaTeX främjar) är att sätta dit en liten pil ovanför variabeln ($\vec{a}$), vilket görs med `\vec`. Andra föredrar att skriva variabeln fetstilt eller med krittavletypsnitt, vilket görs med `\boldmath` och `\mathbb`, respektive.

Min rekommendation är att använda `\vec` (eftersom det i koden indikerar att vi pratar om en vektor), och omdefiniera den om man föredrar en annan notation. Man kan till exempel använda följande kommando för att göra så att `\vec` istället visar variabeln fetstilt:

```
\renewcommand\vec[1]{\boldmath{#1}}
```

Paranteser och dylikt

Paranteser är mycket viktiga inom matematiken för att visa olika typer av grupper, och besitter dessutom egenskapen att de måste skalas med sin omgivning för att de ska se bra ut. I LaTeX skapar man sådana grupper av paranteser med kommandona `\left` och `\right`, vilket gör att dessa paranteser automatiskt sträcks ut för att bli lika höga som innehållet i gruppen (vilket visas i exempel 7 på sidan 36). Man använder `\left` och `\right` genom att sätta dit parantesen direkt efter kommandot:

```
\left(\frac{1}{2}\right)
```

Tabell 9: *Viktiga pilar i matematisk notation och deras LaTeX-kommandon.*

Kort		Lång	
`\gets`	$\leftarrow$	`\longleftarrow`	$\longleftarrow$
`\to`	$\rightarrow$	`\longrightarrow`	$\longrightarrow$
`\leftrightarrow`	$\leftrightarrow$	`\longleftrightarrow`	$\longleftrightarrow$
`\Leftarrow`	$\Leftarrow$	`\Longleftarrow`	$\Longleftarrow$
`\Rightarrow`	$\Rightarrow$	`\Longrightarrow`	$\Longrightarrow$
`\Leftrightarrow`	$\Leftrightarrow$	`\iff`	$\Longleftrightarrow$
`\mapsto`	$\mapsto$	`\longmapsto`	$\longmapsto$

Tabell 10: *Viktiga relationssymboler och deras LaTeX-kommandon.*

Mindre		Större		Lika	
`<`	$<$	`>`	$>$	`=`	$=$
`\nless`	$\not<$	`\ngtr`	$\not>$	`\neq`	$\neq$
`\leq`	$\leq$	`\geq`	$\geq$	`\equiv`	$\equiv$
`\nleq`	$\not\leq$	`\ngeq`	$\not\geq$		
`\lesssim`	$\lesssim$	`\gtrsim`	$\gtrsim$	`\approx`	$\approx$
`\ll`	$\ll$	`\gg`	$\gg$		
`\subset`	$\subset$	`\supset`	$\supset$		
`\subseteq`	$\subseteq$	`\supseteq`	$\supseteq$		

Tabell 11: *Viktiga operatorer och deras LaTeX-kommandon.*

Operator		Kommentar
`+`, `-`	$+$, $-$	
`\times`, `\div`	$\times$, $\div$	Multiplikation kan även skrivas ut tydligt med `\cdot` ($\cdot$); `\times` och `\cdot` har speciella betydelser inom linjär algebra
`\pm`, `\mp`	$\pm$, $\mp$	
`\cup`, `\cap`	$\cup$, $\cap$	Union/snitt i mängdlära
`\lor`, `\land`	$\vee$, $\wedge$	Och/eller i Boolesk algebra
`\partial`	∂	Partiell deriveringsoperator
`\nabla`	∇	Flerdimensionell deriveringsoperator
`\Delta`	Δ	Laplaceoperator

Tabell 12: *Fyra vanliga parantestyper i LaTeX.*

Paranteser		Kommentar
`\{ \}`	{ }	Används för bland annat mängddefinitioner
`( )`	( )	
`[ ]`	[ ]	Används av vissa istället för (), av andra för att visa insättning av en variabel
`\langle` `\rangle`	⟨ ⟩	Används för att visa skalärprodukten i till exempel funktionsrum

De paranteser som oftast används listas i tabell 12. Värt att notera är att `\left` och `\right` måste användas i par, just eftersom de skapar en grupp. Vill man bara använda en av dem måste man ändå sätta dit den andra på en lämplig plats, men med en punkt efter istället för en parantes. Punkten motsvarar tomrum:

```
\left.\frac{1}{2}\right]
```

Paranteserna måste alltså inte heller matcha typmässigt, även om detta ofta är önskvärt:

```
\left(\frac{1}{2}\right]
```

Att ha omatchade paranteser kan vara önskvärt inom till exempel kvantfysik, där man har braket-notation (se dock notisen om `\mid` på motstående sida och paketet braket som förklaras på sidan 62):

```
\left\langle\psi\right\vert
```

```
\begin{equation*}
  \int\limits_0^1\!\left(\dfrac{x^3}{3} +
  x^2\right)\d{x} = \left[\dfrac{x^4}{12} +
  \dfrac{x^3}{3}\right]_{x=0}^1 = \dfrac{5}{12}
\end{equation*}
```

$$\int_0^1 \left(\frac{x^3}{3} + x^2\right)\,\mathrm{d}x = \left[\frac{x^4}{12} + \frac{x^3}{3}\right]_{x=0}^1 = \frac{5}{12}$$

Exempel 7: *Skalbara paranteser med `\left` och `\right`.*

Tabell 13: *Sum-, produkt- och integraltecken i LaTeX.*

`\sum`	$\sum$	`\prod`	$\prod$	`\coprod`	$\coprod$
`\int`	$\int$	`\iint`	$\iint$	`\iiint`	$\iiint$
`\iiiint`	$\iiiint$	`\idotsint`	$\int\cdots\int$	`\oint`	$\oint$

Vertikala streck

Vertikala streck är mycket lika paranteser; man kan även använda dessa genom att skriva exempelvis `\left|` och `\right|` (eller `\left\vert` och `\right\vert`), om man vill. Det finns dock även kommandon `\lvert` och `\rvert` som motsvarar dessa (och som beter sig något bättre typsättningsmässigt), samt `\lVert` och `\rVert` som motsvarar dubbla vertikala streck. Man kan således definiera absolutbelopps- och normkommandon i LaTeX[1][2]:

```
\DeclarePairedDelimiter\abs{\lvert}{\rvert}
\DeclarePairedDelimiter\norm{\lVert}{\rVert}
```

Dessa kan sedan enkelt användas i matematikläge:

```
\norm{x}_1 = \sum_{i=1}^n\abs{x_i}
```

Dessutom används det vertikala strecket ibland mitt i block (exempelvis vid definition av mängder, $\varnothing = \{A \mid A \in A\}$) och betecknas då `\mid`. Ett exempel med tidigare nämnda bra-ket-notation kan se ut så här:

```
\left\langle\psi\mid\Psi\right\rangle
```

Dessutom kan det användas som en operator, exempelvis för att visa insättning av en variabel som ett led i integrering ($\int_1^e x^{-1}\,\mathrm{d}x = \ln x|_{x=1}^e = 1$) i vilket fall man istället skriver `\vert`:

```
\ldots=\ln x\vert_{x=1}^e=\ldots
```

Summor och integration

Integraler och summor (och upprepade produkter och så vidare) är mycket enkla att uttrycka med LaTeX. Med hjälp av kommandot `\sum` (eller `\int`, se även tabell 13) kan man infoga en summa (eller integral) utan gränser. Vill man ha en definit integral, eller en summa med gränser, så använder man sub- och superskriptskommandona `_` och `^` som tidigare diskuterats på sidan 30:

```
\sum_{x\in A}\left(\ldots\right)
```

[1] Tack till Lev Bishop för grunden till dessa definitioner (http://bit.ly/lev-bishop-abs-norm). [2] Kommandot `\DeclarePairedDelimiter` definieras av paketet mathtools

Detta resulterar oftast i ett korrekt resultat, med gränser över och under tecknet då detta får plats, och med gränser likt vanliga upphöjningar och nedsänkningar annars (till exempel i ekvationer i text, där det är ont om plats i höjdled). Ibland gör LaTeXdock fel. När detta händer får man hinta LaTeX om att gränserna inte typsätts korrekt med hjälp av kommandot `\limits`, som placeras mellan tecknet och gränserna:

```
\sum\limits_{x=1}^\infty \frac{1}{x^2}
```

Kommandot `\limits` kan även användas med andra operatorer, till exempel max, inf och lim:

```
\lim\limits_{x\to\infty} \frac{1}{x}
```

Snyggare (och mer korrekta) integraler

Integraler, så som LaTeX typsätter dem från början, blir ofta felaktiga i det avseendet att avståndet mellan integraltecken och innehåll blir stort, samtidigt som avståndet till differentieringsoperatorn (som dessutom inte typsätts korrekt utan visst krångel) blir för litet. Det första problemet (som i viss mån även finns för summor) kan åtgärdas genom att lägga in ett negativt mellanrum (`\!`) mellan integraltecken och innehåll:

```
\int_0^1\! x^2 dx
```

Det andra problemet kan också lösas genom att lägga till mellanrum (i kombination med att typsätta *dx* med `\mathrm`), men eftersom det är så ofta man behöver differentieringsoperatorn är det bättre att definiera ett nytt kommando för detta[1]:

```
\makeatletter
\renewcommand\d[1]{\ensuremath{%
  \;\mathrm{d}#1\@ifnextchar\d{\!}{}}}
\makeatother
```

Med denna definition kan man mycket enklare typsätta snygga integraler med korrekta differentieringsoperatorer, vilket exempel 8 på motstående sida visar. Notera även att exemplet inte använder `\iint` trots att det handlar om en dubbelintegral; detta eftersom varje ”underintegral” har egna gränser. Integraltecknet `\iint` används istället med fördel när man ska visa integrering över exempelvis ett område *A*, då detta bara är ”en” gräns:

```
\iint_A\! f(x) \d{A}
```

Ekvationsomgivningar

Redan på sidan 30 diskuterades två av de ekvationsomgivningar AMSLaTeX erbjuder (`equation` och `equation*`), och dessa två är fullt tillräckliga för väldigt många tillämpningar. Det finns dock tillfällen då dessa inte räcker till, och därför definierar AMSLaTeX även en stor mängd andra omgivningar för att lösa de problem som kan uppstå. Det kan handla om att man har en väldigt

[1] Tack till Niel de Beaudrap för grunden till detta kommando (http://bit.ly/beaudrap-dx).

$$\int_0^\infty \int_0^1 f(x,y) dy dx$$

```
\begin{equation*}
  \int\limits_0^\infty
  \int\limits_0^1 f(x,y) dydx
\end{equation*}
```

$$\int_0^\infty \int_0^1 f(x,y)\,\mathrm{d}y\,\mathrm{d}x$$

```
\begin{equation*}
  \int\limits_0^\infty\!
  \int\limits_0^1\! f(x,y) \d{y}\d{x}
\end{equation*}
```

Exempel 8: *Integral med korrekt (understa) och inkorrekt (överst) typsättning. Notera att* `\d` *inte finns i LaTeX från början utan måste definieras med den kod som presenteras på motstående sida.*

lång ekvation som måste brytas över flera rader, att man har flera små ekvationer som hör ihop men som bör vara radbrutna, att man vill ha en matris eller något helt annat.

Den här introduktionen kommer bara att presentera några av de viktigaste omgivningarna; en fullständig lista finns i användarguiden till amsmath (American Mathematical Society 1999) och en ordentlig genomgång ges av Voß (2010).

Långa ekvationer med `multline`

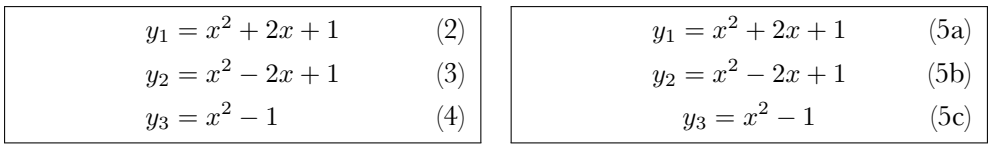
Ibland (oroväckande ofta, till och med) stöter man på ekvationer som är alldeles för långa för att få plats på en rad. Hamnar man i en sådan situation är det `multline` (notera avsaknandet av ett `i`) man bör använda. Omgivningen typsätter matematik på flera rader, där den första raden vänsterjusteras, sista raden högerjusteras och rader däremellan centreras — ett exempel kan ses i (1).

$$\int_0^\infty R(t)\,\mathrm{d}t = -3\sqrt{2\pi e}\,\operatorname{erfc}\left(\frac{1}{\sqrt{2}}\right) - 3e^2\sqrt{2\pi}\left(e^{5/2}\operatorname{erfc}\left(\frac{3}{\sqrt{2}}\right) - 2\operatorname{erfc}\left(\sqrt{2}\right)\right) + 2 \quad (1)$$

Rader separeras helt enkelt med radbrytningskommandot `\\`:

```
\begin{multline}
  \lim_{x\to a} \frac{f(x) - f(a)}{x - a} = \\
  \ldots \\
  = f'(a)
\end{multline}
```

Precis som `equation` finns `multline` både med och utan stjärna, och på exakt samma sätt har varianten utan stjärna ett ekvationsnummer, vilket den med stjärna inte har. I övrigt finns det inga skillnader mellan de båda. Notera att varje `multline`-omgivning endast får *ett* ekvationsnummer; vill man ha ett per rad ska man istället använda `align`.



$$y_1 = x^2 + 2x + 1 \quad (2)$$
$$y_2 = x^2 - 2x + 1 \quad (3)$$
$$y_3 = x^2 - 1 \quad (4)$$

(**a**) *Tre ekvationer typsatta med* `align`

$$y_1 = x^2 + 2x + 1 \quad (5a)$$
$$y_2 = x^2 - 2x + 1 \quad (5b)$$
$$y_3 = x^2 - 1 \quad (5c)$$

(**b**) *Tre ekvationer typsatta med* `gather`

Figur 1: *Relaterade ekvationer typsatta med* `align`*,* `gather` *och* `subequations`*.*

Relaterade ekvationer: `align` och `gather`

Ibland kan man ha flera olika, men relaterade ekvationer man vill typsätta i en grupp eller på rad. Detta kan man göra med två olika omgivningar: `align` och `gather` (även dessa har stjärnvarianter utan nummer). Figur 1 visar hur dessa omgivningar ser ut; den enda skillnaden mellan de två är att man med `align` kan placera ekvationer mer exakt; `gather` centrerar helt enkelt bara ekvationerna.

Nya rader skapas i båda omgivningarna med nyradskommandot `\\`, och i `align` använder man (på liknande sätt som i `tabular`) ett et-tecken (`&`) för att skapa en ”kolumn”. Koden för Figur 1a är således som följer:

```
\begin{align}
  y_1 &= x^2 + 2x + 1 \\
  y_2 &= x^2 - 2x + 1 \\
  y_3 &= x^2 - 1
\end{align}
```

I varianterna utan stjärna får varje rad ett eget ekvationsnummer. Detta är helt i sin ordning; `align` och `gather` ska användas för att gruppera ekvationer, och om man istället vill bryta upp en ekvation på flera rader bör man använda `multline`. Man kan dock tvinga LaTeX att utelämna ekvationsnumret för en specifik rad genom att använda kommandot `\nonumber` precis innan man bryter raden. Man kan även sätta en etikett på en enskild rad genom att placera `\label` på samma sätt.

Underekvationer

Förutom `align` och `gather` visar även figur 1b hur omgivningen `subequations` fungerar. Denna omgivning gör att man kan skapa ”underekvationer” (likt underrubriker), och får bara innehålla en (och endast en) matematikomgivning, inget annat:

```
\begin{subequations}
  \begin{align}
    y_1 &= x^2 + 2x + 1 \\
    y_2 &= x^2 - 2x + 1 \\
    y_3 &= x^2 - 1
  \end{align}
\end{subequations}
```

$$\begin{Vmatrix} a & b \\ c & d \end{Vmatrix} \quad \begin{Bmatrix} a & b \\ c & d \end{Bmatrix} \quad \begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

(**a**) *Vmatrix* (**b**) *Bmatrix* (**c**) *vmatrix* (**d**) *bmatrix* (**e**) *pmatrix*

Figur 2: *Fem av de sex matristyper AMSLaTeX definierar.*

Matriser

Matriser byggs upp på ett sätt som liknar `tabular`, och definieras av de omgivningar som visas i figur 2. Dessa måste användas i matematikläge, det vill säga i en annan omgivning som används för att typsätta matematik. Kolumner separeras med `&` och radbrytningar görs med `\\`.

I stora eller repetitiva matriser (till exempel enhetsmatrisen) behöver man inte alltid skriva ut alla element, utan istället använda notation för upprepande element. Detta görs med hjälp av tre kommandon; `\cdots`, `\vdots` och `\ddots`, som skapar horisontella, vertikala och diagonala punkter vilka kan användas istället för innehåll i kolumnerna i en matris:

```
\begin{equation*}
  \lvert I\rvert = \begin{vmatrix}
    1      & 0      & \cdots & 0      \\
    0      & 1      & \cdots & 0      \\
    \vdots & \vdots & \ddots & \vdots \\
    0      & 0      & \cdots & 1
  \end{vmatrix} = 1
\end{equation*}
```

Olika fall med `cases`

Omgivningen `cases` är mycket användbar då man definierar styckvisa funktioner eller vissa talföljder, till exempel den som behandlas i Collatz problem. Omgivningen liknar både `align` och `matrix`-omgivningarna, men har endast två kolumner (som dock separeras av mer mellanrum än i andra omgivningar). Exempel 9 visar hur `cases` används för att definiera en styckvis funktion (i det här fallet rampfunktionen).

$$R(x) = \begin{cases} x, & x \geq 0; \\ 0, & x < 0 \end{cases} \tag{6}$$

```
\begin{equation}
  R(x) = \begin{cases}
    x, & x \geq 0; \\
    0, & x < 0
  \end{cases}
\end{equation}
```

Exempel 9: *Rampfunktionens definition typsatt med hjälp av `cases`.*

Mellanrum

Mellanrum i matematikläge ignoreras som sagt av LaTeX, som automatiskt sätter in korrekt mellanrum där det behövs. Normalt brukar detta bli mycket bra, men ibland (som i fallet med integraler) kan man behöva göra en del manuella justeringar. Detta kan man göra med de kommandon som ges i tabell 14.

Det finns både positiva (vanliga) och negativa mellanrum. De negativa flyttar inte helt oväntat text ”bakåt”, och är användbara när man till exempel vill ångra de mellanrum LaTeX lägger in.

Vilket mellanrum LaTeX lägger in beror på vilken typ tecknet är och hur omgivningen ser ut; tabell 15 på motstående sida visar de teckentyper som används i matematikläge samt några exempel och mängden mellanrum som tillförs. Notera dock att tecken av typ 2 (binära operatorer) görs om till typ 0 om de inte har någon text (en operand) till vänster.

Fantomer

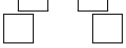
Ibland finns det även tillfällen då man vill ha mellanrum som är *exakt* lika stora som en viss bit typsatt text. Man använder då fantomer. De viktigaste fantomerna är `\hphantom` (ingen höjd, bredd från dess innehåll) och `\vphantom` (ingen bredd, höjd från dess innehåll). Dessa kan vara användbara om man till exempel använder `multline` för att bryta en ekvation, samtidigt som man använder `\left` och `\right` (som inte fungerar ”runt” radbrytningar).

Figur 3 på nästa sida illustrerar problemet och lösningen. Här använder vi `\vphantom`, eftersom vi vill lägga till höjd, och vi fyller den med `\frac{1}{2}` eftersom det är den högsta delen av gruppen vi vill matcha.

Punkter

Även om LaTeX definierar kommandon som `\ldots` och `\cdots` för att typsätta punkter så bör man undvika dessa eftersom de inte berättar vilket sammanhang punkterna används i. Det finns många olika konventioner, och för att enkelt kunna byta mellan dessa är det bättre att använda *semantiska* punktkommandon. AMS LaTeX definierar fem sådana (tabell 16 på motstående sida) för olika tillfällen. Använd alltid dessa om det är möjligt!

Tabell 14: *De mellanrum LaTeX definierar.* (1 mu = 1/18 em)

Kommando	mu	Exempel
`\negthickspace`	−5	
`\negmedspace`	−4	
`\!`, `\negthinspace`	−3	
	0	
`\,`, `\thinspace`	3	
`\:`, `\medspace`	4	
`\;`, `\thickspace`	5	
`\quad`	18	
`\qquad`	36	

$$p(x) = \left(\frac{1}{2}x^3 + x^2 + x + 1\right) = 0 \quad (7)$$

```
\begin{multline}
  p(x) = \left(\frac{1}{2}x^3 +
         x^2 + \right. \\
  \left. x + 1 \right) = 0
\end{multline}
```

$$p(x) = \left(\frac{1}{2}x^3 + x^2 + x + 1\right) = 0 \quad (8)$$

```
\begin{multline}
  p(x) = \left(\frac{1}{2}x^3 +
         x^2 + \right. \\
  \left. x + 1 \vphantom{
  \frac{1}{2} }\right) = 0
\end{multline}
```

Figur 3: *Problem med* `multline` *och lösningen med* `\vphantom`.

Tabell 15: *Avstånd före (**pre**) och efter (**post**) olika typer av tecken i matematikläge.*

Typ		Beskrivning	Avst. (mu) Pre	Post
0	$A0\Phi\infty$	Enkla tecken (”substantiv”)	0	0
1	$\sum\prod\int$	Prefixoperatorer	0	3
2	$+\cup\wedge$	Binära operatorer	4	4
3	$=<\subset$	Jämförelse (”verb”)	5	5
4	$([\{\langle$	Öppnande avgränsare	0	0
5	$)]\}\rangle$	Stängande avgränsare	0	0
6	$.,;!$	Postfix, punktuation	0	0

Tabell 16: *Semantiska punktkommandon definierade av AMS-LaTeX.*

Kommando	Exempel	Kommentar
`\dotsc`	$A_1, A_2, \dots, A_n$	Används i anslutning till kommatecken, det vill säga listor och dylikt (eng. ***dots** with **c**ommas*)
`\dotsb`	$A_1 + A_2 + \cdots + A_n$	Används i anslutning till binära operatorer, alltså addition, subtraktion, booleska operatorer och så vidare (eng. ***dots** with **b**inary operators*)
`\dotsm`	$A_1 A_2 \cdots A_n$	Används i anslutning till multiplikation (eng. ***dots** for **m**ultiplication*)
`\dotsi`	$\int_{A_1}\int_{A_2}\cdots\int_{A_n}$	Används med integraler, summor och liknande (eng. ***dots** with **i**ntegrals*)
`\dotso`	$A_1 \dots A_n$	Används då inga andra punkter passar (eng. ***dots** for **o**ther situations*)

Enheter med siunitx

Fysiker och matematiker (främst fysiker) behöver ofta typsätta enheter i anslutning till sin text, något som inte alltid är helt enkelt eftersom en del konventioner existerar gällande avstånd och så vidare. Paketet siunitx ämnar förenkla detta genom att tillhandahålla kommandon som till exempel `\SI`.

Paketet definierar tre kommandon (`\num`, `\ang` och `\SI`) som används för att typsätta tal, vinklar och tal med enheter, respektive. Det tal man vill typsätta kan även inkludera förenklande notation så som `+-` istället för `\pm`, `e` för att beteckna tiopotenser och så vidare. Även komplexa tal stöds.

Kommandona `\num` och `\ang`, som inte accepterar enheter, kräver endast ett argument medan `\SI` kräver två; talet som ska typsättas och enheten som hör till. Enheterna ges av LaTeX-kommandon så som `\metre`, `\per`, `\milli` och så vidare (en full lista ges av Wright 2011, ss. 9–12):

```
\num{12345,60} % Typsätter ett tal
\num{.35e100} % Typsätter ett stort tal
\num{1+-2i}   % Typsätter ett komplext tal
\ang{180}     % Typsätter en vinkel
\SI{15}{\kilogram\metre\per\second\squared} % Tal med enhet
```

Dessa kommandon typsätts då sedan som $12\,345{,}60$, $0{,}35 \cdot 10^{100}$, $1 \pm 2\mathrm{i}$, $180°$ och $15\,\mathrm{kg\,m\,s^{-2}}$, respektive.

Som en extra bonus löser siunitx även, om man ställt in paketet ordentligt, problemet med tusentals- och decimalseparatorn som LaTeX ofta ger upphov till (eftersom systemet baseras på engelska konventioner). Detta gör man genom att använda `\sisetup` för att sätta `locale` till `DE`[1]:

```
\sisetup{locale=DE}
```

Paketet kan göra väldigt mycket (man kan även definiera sina egna enheter), och det är omöjligt att gå igenom allt i en kort introduktion. Den intresserade hänvisas till siunitx-manualen (Wright 2011), som utförligt förklarar hur paketet fungerar och vilka inställningar som kan göras.

[1] Tyska konventioner väljs eftersom paketet inte innehåller inställningar för svenska, och eftersom tyska konventioner är mycket nära de svenska.

IV
Grafik med LaTeX

Många dokument kräver att man inkluderar figurer för att presentera resultat eller förtydliga resonemang. Dessa typsätts i LaTeX med hjälp av flytande objekt, vilket diskuteras i del II, men i dessa flytande objekt måste man även importera eller generera sin grafik.

LaTeX ger genom diverse paket många möjligheter att göra detta. Med den klassiska LaTeX-motorn, som genererar DVI-filer, kunde man importera EPS-filer eller rita direkt med PostScript-kommandon, men eftersom man i dagsläget oftast använder (och bör använda) pdfLaTeX måste man använda något annorlunda metoder. Denna del förklarar två av dessa: graphicx och PGF/Ti*k*Z.

Inkludera grafik med graphicx

Det enklaste sättet att inkludera grafik är att skapa sina figurer i ett externt program (det kan till exempel vara fördelaktigt att exportera sina figurer direkt från den programvara man använder) och sedan importera dessa till LaTeX-dokumentet. Detta gör man med paketet graphicx.

Proceduren är enkel:

1. Skapa figuren i ett externt program och spara i rätt format
2. Inkludera graphicx-paketet genom att skriva `\usepackage{graphicx}` i inledningen till dokumentet.
3. Inkludera grafiken i en `figure`-omgivning med hjälp av kommandot `\includegraphics`:

   ```
   \includegraphics[parametrar]{filnamn}
   ```

   Parametrarna här kan vara många. Oftast vill man ställa in bredden på sin bild så att den passar sidan, och detta kan man göra genom att specificera parametern `width` och sätta den till `\textwidth`:

   ```
   \includegraphics[width=\textwidth]{filnamn}
   ```

   Andra parametrar som kan användas är `height`, `angle` och `trim`[1]. En komplett beskrivning av paketet och de parametrar som finns att tillgå finns i dokumentationen på CTAN[2].

[1] Man bör då inte heller glömma att sätta `clip=true`.

[2] http://www.ctan.org/archive/macros/latex/required/graphics/grfguide.pdf

```
\begin{figure}
  \centering
  \includegraphics
    [width=.25\textwidth]
    {interpolation.png}
  \caption{Ett exempel på
    interpolation}
\end{figure}
```

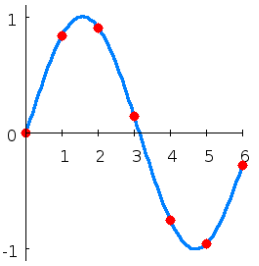


Figure 1: Ett exempel på interpolation

Figur 4: *Koden till vänster inkluderar filen* `interpolation.png` *i LaTeX-dokumentet och ger det resultat som ses till höger.*

Säg att man till exempel vill inkludera filen `interpolation.png` i sitt dokument för att illustrera konceptet interpolation. Man bestämmer sig dock för att figuren bara ska ta upp 25 % av textbredden, eftersom figuren bli för stor annars. Man använder då parametern `width` enligt figur 4.

Rätt format?

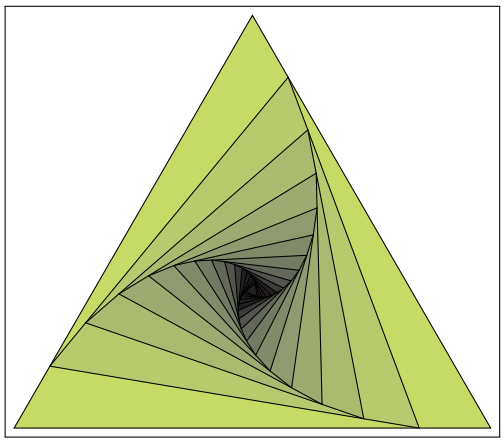
En nackdel med pdfLaTeX när det gäller graphicx är att det inte finns särskilt många bildformat som fungerar (dock fler än för gamla LaTeX, som bara accepterade **EPS**). De format som kan inkluderas med graphicx när man använder pdfLaTeX är **PNG**, **JPEG** och **PDF**, och oftast vill man använda **PDF** för figurer eftersom det är det enda vektorbaserade formatet som fungerar, medan man för fotografier och dylikt använder **JPEG**. **PNG** kan man använda då man egentligen bör använda **PDF** men detta inte är möjligt.

Eftersom många verktyg och programvaror fortfarande sparar filer i **EPS**-format kan det vara praktiskt att kunna konvertera dessa till ett format som fungerar. Detta kan göras med verktyget `epstopdf`, som finns tillgängligt på CTAN.

Rita med PGF/TikZ

Ett lite mer komplicerat sätt att inkludera grafik (men ofta föredelaktigt, eftersom allt innehåll stannar i TeX-filen) är att använda PGF/TikZ, ett paket skrivet för att användas med pdfLaTeX för att rita figurer i LaTeX. Med det kan man direkt i sitt LaTeX-dokument skapa enklare figurer som flödesscheman, träd, grafer eller liknande; se exempel i figur 5 på motstående sida. Även mer avancerade figurer är möjliga, men att förklara hela PGF/TikZ tar allt för lång tid för denna introduktion. Den intresserade läsaren hänvisas istället till Mertz och Slough (2007) som har en bra introduktion till ämnet, och PGF/TikZ-manualen (Tantau 2010) som utförligt beskriver hur paketet fungerar.

```
\newcounter{d}\setcounter{d}{0}
\def\mcolor{SpringGreen}
\begin{tikzpicture}
    \path[coordinate] (0,0)
         coordinate(A) ++( 60:6cm)
         coordinate(B) ++(-60:6cm) coordinate(C);
    \draw[fill=Black!\thed!\mcolor]
         (A) -- (B) -- (C) -- cycle;
    \foreach \x in {1,...,15}{%
        \pgfmathsetcounter{d}{\thed+10}
        \setcounter{d}{\thed}
        \path[coordinate] coordinate(X) at (A){};
        \path[coordinate] (A)
             -- (B) coordinate[pos=.15](A)
             -- (C) coordinate[pos=.15](B)
             -- (X) coordinate[pos=.15](C);
        \draw[fill=Black!\thed!\mcolor]
             (A)--(B)--(C)--cycle;
    }
\end{tikzpicture}
```



Figur 5: *Koden ovan tolkas av PGF/TikZ och resulterar i den itererade triangeln till vänster. Fler exempel finns i Mertz och Slough (2007) och Tantau (2010). Tack till Alain Matthes för originalkoden*[a].

[a] http://www.texample.net/tikz/examples/rotated-triangle/

V
Referenser med BibTeX

En stor fördel med LaTeX är att man kan automatisera hanteringen av till exempel referenser. BibTeX är ett verktyg som gör detta ännu enklare och som gör att man kan samla alla sina referenser i en extern databas, vilken kan hanteras av ett externt program[1] eller (som sig bör när det gäller LaTeX) med en enkel textredigerare.

Systemet består av stilfiler (med filändelse `.bst`), databaser (`.bib`) och programmet `bibtex`. Denna introduktion kommer endast att gå igenom databasformatet och hur denna används från LaTeX; en ytterligare introduktion ges av Fenn (2006) och en komplett manual är även tillgänglig (Patashnik 1988b). Att skapa egna stilfiler involverar en del programmering i ett stackorienterat postfix-språk som dokumenteras av btxhak (Patashnik 1988a), men detta är inte att rekommendera. Bättre är att använda några av de BibTeX-stilar som finns på CTAN (till exempel chscite, om man vill följa Chalmers-bibliotekets rekommendationer).

BibTeX-databasen

Den databas BibTeX använder är i all sin enkelhet en vanlig textfil. Denna textfil innehåller ett antal block, ett per referens i databasen, som innehåller information om referensen (*fält*). En typisk referens innehåller en nyckel (vilken man sedan använder när man refererar till referensen i LaTeX-dokumentet), en titel, en författare och ett årtal. Dessutom innehåller varje block information om vilken typ av referens (bok, artikel och så vidare) det handlar om.

Exempel 10 på nästa sida visar ett block ur en BibTeX-databas; blocket inleds med en blocktyp (`@ARTICLE`) och den nyckel som används då man refererar till källan från LaTeX (`Hassanpour08`) varefter ett antal självförklarande fält med information följer. Olika blocktyper kräver olika fält, både obligatoriska (till exempel titel och författare) och frivilliga (i det här fallet är bland annat `month` frivillig) — tabell 17 på följande sida listar de vanligaste blocktyperna och de obligatoriska/frivilliga fält som hör till.

Oftast är det dock lättare att hantera sin referensdatabas med någon form av dedikerat verktyg istället för att redigera `.bib`-filerna för hand med en textredigerare. Programmet JabRef[2] fungerar mycket bra till detta och finns tillgängligt för alla plattformar.

Många webbaserade sökverktyg för akademiska publikationer, till exempel Scopus[3], gör det möjligt att enkelt exportera information om de artiklar man refererar till i BibTeX-format, som sedan kan läggas till direkt i den egna referensdatabasen. Detta är mycket praktiskt i större projekt så som kandidatuppsatser.

[1] Exempel på sådana är *JabRef*, *BibDesk* och *refbase*.

[2] http://jabref.sourceforge.net/

[3] http://www.scopus.com/home.url

```
@ARTICLE{Hassanpour08,
  author = {Hassanpour, R. and Shahbahrami, A. and
            Wong, S.},
  title = {Adaptive Gaussian Mixture Model for Skin
           Color Segmentation},
  journal = {World Academy of Science,
             Engineering and Technology},
  year = {2008},
  volume = {41},
  pages = {1--6},
  month = may
}
```

Exempel 10: *En enkel exempelreferens ur en* BibTeX*-databas.*

Tabell 17: *Vanliga referenstyper i* BibTeX *och deras tillhörande fält. Både typnamn och fältnamn är relativt självförklarande.* ***Nyckel:*** *obligatoriskt, valfritt och otillgängligt fält.*

Blocktyp	address	author	booktitle	chapter	edition	editor	journal	month	number	pages	publisher	school	series	title	volume	year
@article	■	■	■	■	■	■	■	□	□	□	■	■	■	■	□	■
@book[a]	□	■	■	■	□	■	■	□	■	■	■	■	□	■	□	■
@booklet	□	□	■	■	■	■	■	□	■	■	■	■	■	■	■	□
@inbook[ab]	□	■	■	■	□	■	■	□	■	■	■	■	□	■	□	■
@incollection	□	■	■	■	□	■	■	□	■	□	□	■	■	■	■	■
@inproceedings	□	■	■	■	□	■	■	□	■	□	□	■	■	■	■	■
@manual	■	□	■	■	□	■	■	□	■	■	■	■	■	■	■	□
@mastersthesis	□	■	■	■	■	■	■	□	■	■	■	■	■	■	■	■
@misc	■	□	■	■	■	■	■	□	■	■	■	■	■	□	■	□
@phdthesis	□	■	■	■	■	■	■	□	■	■	■	■	■	■	■	■
@proceedings	□	■	■	■	■	□	■	□	■	■	□	■	■	■	■	■
@unpublished	■	■	■	■	■	■	■	□	■	■	■	■	■	■	■	□

[a] Antingen `author` *eller* `editor` ska specificeras, ej båda två.
[b] Det räcker med att specificera ett av fälten `chapter` och `pages`.

BibTeX med LaTeX

För att referera till de referenser man har i sin referensdatabas i sitt LaTeX-dokument krävs ett par olika saker. Först och främst så måste man specificera en bibliografistil och inkludera sin databas i dokumentet där man vill ha sin bibliografi:

```
\bibliographystyle{plain}
\bibliography{databas}
```

Notera här att bibliografistilen (mer om dessa på nästa sida) måste specificeras innan databasen inkluderas, och att databasen inkluderas med sitt namn *utan filändelsen*; koden ovan inkluderar alltså referenser från filen `databas.bib`. Man kan specificera flera olika databaser genom att separera dem med kommatecken.

När man sedan kompilerar sitt dokument måste man även köra BibTeX på sitt dokument; detta görs genom att köra kommandot `bibtex "dokument"` (om TeX-dokumentet heter `dokument.tex`) — notera avsaknaden av filändelse även här. Därefter måste man köra LaTeX ännu en gång för att referenserna ska dyka upp. En normal kompileringsrunda med BibTeX är alltså LaTeX→BibTeX →LaTeX→LaTeX. Med extra verktyg så som `latexmk` görs detta automatiskt.

BibTeX inkluderar dock endast de referenser som används i dokumentet (det vill säga sådana som refereras till direkt i texten), så om din referenslista inte dyker upp ska du inte vara orolig. Det finns ett antal olika sätt att referera till källor med BibTeX; två stycken inbyggda (`\cite` och `\nocite`) samt de som definieras av olika paket (till exempel harvard, vilket diskuteras på följande sida).

Kommandot `\cite` refererar till en källa i databasen och skriver ut en hänvisning till bibliografin. För de ursprungliga stilarna i BibTeX är denna hänvisning en siffra eller nyckel inom hakparanteser (alltså något i stil med ”[1]”), men detta kan modifieras av diverse paket och bibliografistilar. Kommandot tar ett obligatoriskt argument, nyckeln till källan i databasen, och ett frivilligt som lägger till text efter hänvisningen. Det frivilliga argumentet kan alltså användas för att hänvisa till specifika sidor eller delar av en källa:

```
\cite{Hassanpour08}       % Skriver ut ”[1]”
\cite[s.~5]{Hassanpour08} % Skriver ut ”[1, s. 5]”
```

Notera här att det obligatoriska argumentet ska motsvara den nyckel som anges i databasen; här refererar vi till den källa som visas i exempel 10 på föregående sida. Man kan även referera till flera källor samtidigt genom att separera dessa med kommatecken:

```
\cite{Hassanpour08,Khan10} % Skriver ut ”[1,2]”
```

Kommandot `\nocite` gör samma sak som `\cite` men skriver inte ut någon hänvisning. Detta kommando kan därför användas om man vill inkludera en källa i bibliografin utan att faktiskt referera till den i text. Här finns endast det obligatoriska argumentet. Vill man inkludera alla källor i databasen i sitt dokument kan man istället för en nyckel ange ”*”:

```
\nocite{*}
```

Bibliografistilar

Det finns en uppsjö olika sätt att referera till källor när man skriver vetenskapliga rapporter, och dessa kan variera kraftigt i stil. Några använder siffror (IEEE, Vancouver) eller nycklar (AMS) för att hänvisa till källor i text medan andra (Harvard, Chicago, APA) använder sig av författarnamnen. BibTeX, i sitt grundutförande, kan endast skapa bibliografier och referenser med nycklar eller siffror. Vissa paket så som harvard och chscite implementerar dock Harvard- och Chicago-stilen.

Även själva bibliografin, alltså listan över källor, kan utformas på många sätt. Figur 6 på motstående sida visar några av de stilar som finns tillgängliga med BibTeX och på CTAN. Vilken stil man vill använda är upp till författaren (eller kanske oftare förläggaren), men generellt sett så brukar man i matematiska kretsar använda `alpha.bst`, i (elektro)ingenjörskretsar `ieeetr.bst` (som ej visas i figuren) och i humanistiska vetenskaper Harvard-stilen (eller någon närliggande stil). Chalmers avviker från detta och rekommenderar att man använder en Chicago-liknande stil (Chalmers Bibliotek 2010).

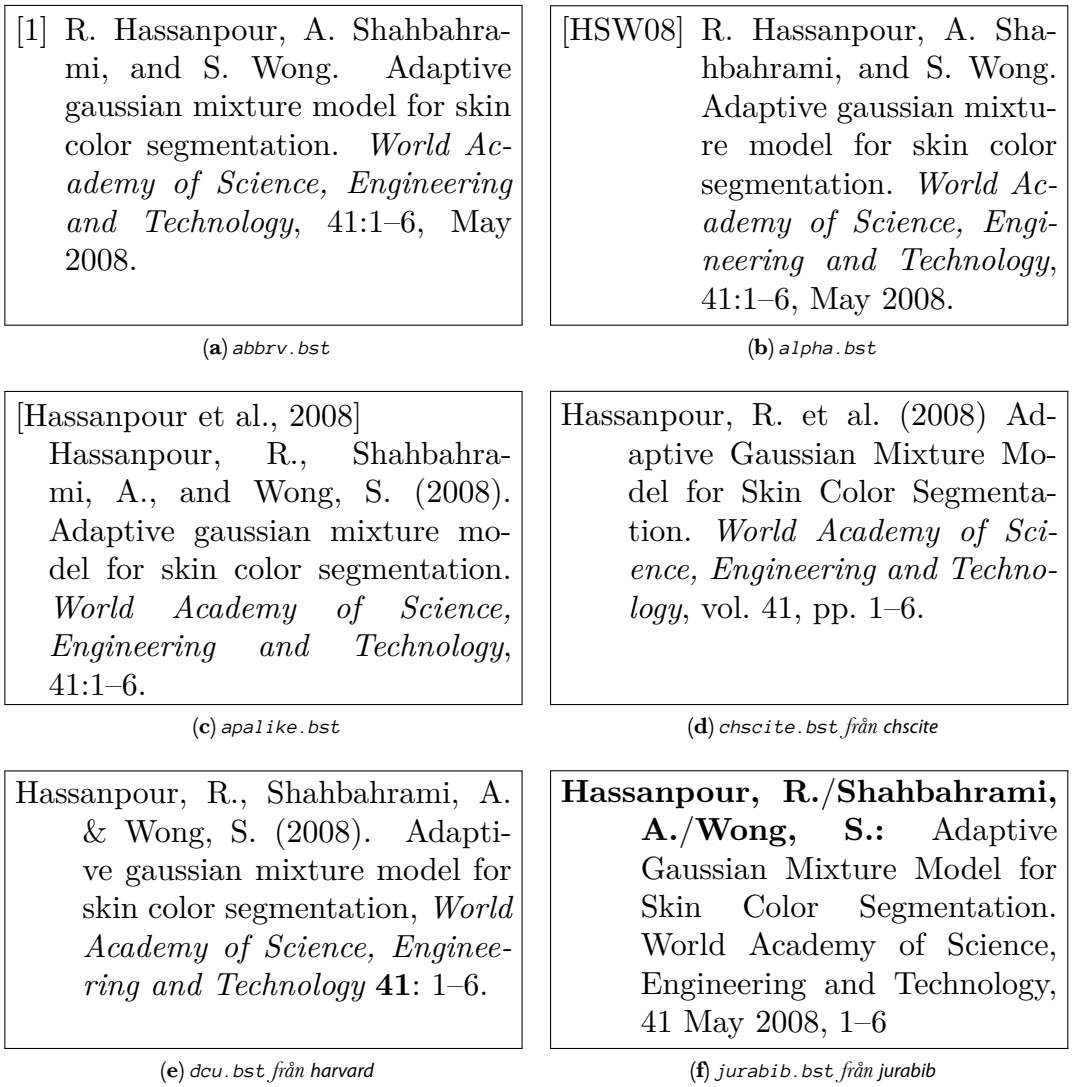
Harvard- och Chicago-stilen: harvard och chscite

Harvard- och Chicago-stilarna, varav den senare beskrivs utförligt i *The Chicago Manual of Style* (Chicago Editorial Staff 2010), är sätt att referera till källor baserat på deras författare, och används ofta i de humanistiska vetenskaperna. Fördelen med dessa stilar är att referenserna smälter in bättre med den omgivande texten och ger ett bättre sammanhang.

Tyvärr så är BibTeX från början inte gjort för att stödja de här referensstilarna, varför det har dykt upp ett antal olika paket som gör det möjligt. Ett av dessa paket är harvard, som beskrivs utförligt av Williams och Schnier (2008). Paketet definierar förutom det vanliga kommandot `\cite` tre relativt självförklarande kommandon: `\citeasnoun`, som refererar till en källa som ett substantiv, `\possessivecite`, som refererar till en källa i genitivform, och `\citeaffixed`, som refererar som `\cite` men med ett tillägg. Hur dessa bör användas i text illustreras av följande exempel:

```
...vilket visades av \citeasnoun{Hassanpour08}.
\possessivecite{Hassanpour08} undersökning visade att...
...flera rapporter \citeaffixed{Hassanpour08}{t.ex.}...
```

Chalmers rekommenderar att man använder en stil baserad på Chicago-stilen (Chalmers Bibliotek 2010), och dessa rekommendationer implementeras av paketet chscite som bygger på harvard och därför fungerar på samma sätt. Även detta paket finns på CTAN.



Figur 6: *Några av de bibliografistilar som kan åstadkommas med hjälp av* BibTeX. *Figurerna (a)–(c) visar standardstilar, medan figurerna (d)–(f) visar stilar som definieras av olika paket.*

VI
Vidare läsning

Den här introduktionen har förhoppningsvis gett dig en bra LaTeX-grund som låter dig typsätta både rapporter och artiklar utan problem. Tyvärr kommer du, eftersom LaTeX är ett så stort system, sannolikt att behöva ytterligare hjälp, tips och resurser allt eftersom du använder LaTeX och stöter på problem eller svårigheter.

Denna del av introduktionen ämnar ge dig några tips på sådana resurser. Delen inleds med några relevanta resurser så som böcker och journaler som behandlar TeX och LaTeX på en mer eller mindre avancerad nivå, samt några tips när det gäller att få hjälp med specifika problem. Därefter följer några tips gällande stora projekt, som du sannolikt kommer att ha nytta av förr eller senare, speciellt om du studerar på en högskola eller ämnar författa böcker och liknande.

Därefter följer en lista av viktiga och nyttiga LaTeX-paket, som man bör åtminstone skumma igenom för att få en hyfsad uppfattning om vilka paket som finns och när man bör använda dem. Avslutningsvis ges även en lista över andra projekt som liknar LaTeX, till exempel det tidigare nämnda XeLaTeX.

Andra resurser

Även om denna introduktion siktar på att ge dig allt du behöver för att lära dig LaTeX är det inte säkert att den är tillräcklig. Man kan inte diskutera allt i en kort introduktion, och vill man lära sig mer om den inre strukturen hos TeX eller LaTeX så finns det redan mycket bra och utförliga resurser tillgängliga. Dessutom kan man i en kort introduktion inte diskutera specifika problem i detalj, eftersom dessa ofta beror på den specifika situationen. Nedan följer därför en lista över resurser i form av böcker, journaler och forum som kan hjälpa dig släcka din kunskapstörst eller lösa dina LaTeX-problem.

Böcker och artiklar

Det finns mycket material tillgängligt när det gäller LaTeX. Många böcker inom ämnet har publicerats av Addison-Wesley, och det finns även en uppsjö artiklar, böcker och guider tillgängliga på CTAN, och en journal (*The PracTeX Journal*[1]) som finns tillgänglig på internet. Nedan följer en lista på några av de böcker och artiklar som kan vara av intresse för dig som precis börjat med LaTeX.

An essential guide to LaTeX 2ε usage **(Fenn och Trettin 2007)** brukar också refereras till som l2tabu (dess namn på CTAN), och ger en kort lista över utdaterade paket, dödssynder inom LaTeX samt några små tips. Förhoppningsvis lär du dig inget nytt av att läsa l2tabu (det skulle

[1] http://tug.org/pracjourn/

ju indikera att den här introduktionen är felaktig), men det kan vara värt att skumma igenom den ändå.

The Not So Short Introduction to LaTeX 2ε **(Oetiker et al. 2011)** går även under namnet lshort och är en kort introduktion till LaTeX, skriven på engelska. Den går *lite* djupare in på vissa bitar av LaTeX, och fokuserar en del på den nu i princip överflödiga LaTeX-motorn som skapar DVI-filer, men har varit en stor inspiration till den här introduktionen. Absolut värd en (snabb) genomläsning.

Math into LaTeX **(Grätzer 1996)** ger än ännu längre introduktion till LaTeX än lshort, och är även den skriven på engelska. En bra ytterligare referens om det är något man vill veta mer om eller något man tycker är otydligt i den här introduktionen och lshort.

Math Mode **(Voß 2010)** ger en mycket ordentlig genomgång av matematiktypsättning både i vanliga LaTeX och med AMS-LaTeX, och är en nästintill oumbärlig referens när man skriver lite mer komplicerade ekvationer. Bokmärk och titta igenom varje gång du skriver matematik i LaTeX.

Short Math Guide for LaTeX **(Downes 2002)** är en kort guide till matematik med AMS-LaTeX skriven av en av huvudpersonerna bakom paketsamlingen. Ger några små värdefulla tips, en lista över symboler och en introduktion till `\DeclareMathOperator` och några andra AMS-konstruktioner som inte diskuteras utförligt i den här introduktionen. Sjutton mycket läsvärda sidor.

The LaTeX Companion **(Mittelbach et al. 2004)** beskriver alla LaTeX-kommandon och en stor mängd paket. Om du ska skriva ett eget paket eller en egen dokumentklass, eller bara är intresserad av att ”hacka” LaTeX lite, så bör du absolut ta en titt på den här boken. En mycket bra referens för den vane TeXaren.

LaTeX: A Document Preparation System **(Lamport 1994)** skrevs av författaren till LaTeX och kan anses vara det närmsta en ”manual” till LaTeX man kan komma. Främst av historiskt intresse, och inget för nybörjaren.

TeX by Topic **(Eijkhout 1992)** ger en utförlig förklaring till TeX och är en relevant referens för alla som ska utföra lågnivåarbete i TeX eller LaTeX. Främst riktad till mycket vana användare av TeX och LaTeX, och absolut inte riktad till nybörjaren.

The TeXbook **(Knuth 1986)** skrevs av skaparen av TeX och förklarar i detalj hur systemet fungerar. Främst av historiskt intresse, och egentligen bara värd att titta på om man vill veta hur TeX *egentligen* fungerar.

Utöver dessa bör man givetvis även läsa manualerna till de paket man använder (dessa finns alltid på CTAN) och kanske även manualen till BibTeX (Patashnik 1988b) om man behöver det.

Hjälp med specifika problem

Det tar lång tid att bemästra LaTeX fullt ut, och i början kommer man garanterat att stöta på problem. Några av dem kanske går att lösa med den hjälp som ges i den här introduktionen och de andra böcker och artiklar som presenteras, men vissa måste man fråga någon om.

Det finns så klart en uppsjö olika maillistor, forum och sökmotorer (och phaddrar, för dig som går på en teknisk högskola) som kan användas för att lösa problem, ställa frågor och utforska LaTeX. En mycket bra resurs är *TeX Stack Exchange*[1] där man kan ställa frågor om TeX, LaTeX och andra

[1] http://tex.stackexchange.com/

```
.
`-- FFM233-projekt
    |-- img
    |   |-- degradering-1.png
    |   |-- degradering-2.png
    |   |-- ...
    |   |-- triangel-1.png
    |   `-- triangel-2.png
    |-- kod
    |   |-- diffekv.m
    |   |-- ode.m
    |   `-- plot.m
    |-- projekt.tex
    `-- referenser.bib
```

Exempel 11: *En bra mappstruktur för ett enkelt LaTeX-projekt.*

relaterade system. Många stora namn i TeX-världen dyker upp där lite då och då, och undrar man något om LaTeX så är det ett utmärkt ställe att fråga. Använder man IRC kan man också med fördel besöka `#latex`-kanalen på Freenode[1].

Tips för stora projekt

För stora LaTeX-projekt (till exempel kandidatarbeten, examensarbeten och liknande) är det viktigt att kunna ha en ordentlig ordning på sitt dokument. Är det dessutom ett dokument många ska samarbeta med är det ännu viktigare att det är strukturerat.

Den första tumregeln, som även bör tillämpas vid mindre projekt, är att man bör lägga varje LaTeX-dokument (eller projekt) i en egen undermapp. Ännu bättre blir det om man även lägger alla externa filer (figurer, inkluderad programkod och så vidare) i ännu en undermapp. En bra mappstruktur skulle alltså kunna se ut ungefär som i exempel 11.

Utöver detta kan man se till att försöka abstrahera bort till exempel kommandodefinitioner eller stiländringar till ett eget paket eller en egen dokumentklass, om möjligt. Att göra detta lämnas som en övning åt läsaren, men mer information om hur man gör sådant ges av bland annat Flynn (2006), The LaTeX3 Project (1999) och Robertson (2006).

Versionshantering

För större projekt och projekt som utförs i grupp är det mycket praktiskt att kunna spåra ändringar och ändra dokumentet från många olika platser (gärna samtidigt). Med hjälp av ett versionshanteringssystem så som Subversion[2] blir detta enkelt. Ännu bättre blir det om man använder ett distribuerat system så som Mercurial[3] eller Git[4], hostat helt gratis av något snällt företag, till exempel på Bitbucket[5] (Github[6] eller Gitorious[7] om man föredrar Git framför Mercurial).

[1] irc://chat.freenode.net [2] http://subversion.tigris.org/ [3] http://mercurial.selenic.com/ [4] http://git-scm.com/
[5] https://bitbucket.org/ [6] https://github.com/ [7] http://gitorious.org/

Att utförligt förklara hur Mercurial fungerar är utanför den här bokens område, men är man intresserad bör man läsa *Hg Init*[1], som trots sitt bruk av Comic Sans är en utmärkt resurs för den som vill lära sig Mercurial och versionshantering.

Uppdelning av dokumentet

Stora projekt (med flera kapitel eller stora delar) kan med fördel delas upp på flera filer. Man har då en grundfil (säg, `projekt.tex`) som refererar till flera olika underfiler (kanske en per kapitel, lämpligen placerade i en undermapp till projektet). Dessa kan man sedan inkludera i grundfilen på ett antal olika sätt, varpå man bara kompilerar

Den första metoden för att inkludera underfilerna är `\input`, som i princip lägger in koden från underfilen direkt där kommandot används och fortsätter kompilera som om koden alltid funnits där. Detta är praktiskt om man av någon anledning inte vill använda den andra metoden.

Den andra metoden, `\include`, är i princip ekvivalent till `\input` med den skillnaden att en sidbrytning läggs in före och efter den inkluderade koden. Dessutom kan man med hjälp av kommandot `\includeonly` bestämma *vilka* underfiler som inkluderas, för att till exempel spara tid vid kompileringen om man bara är intresserad av en specifik del.

Den tredje och sista metoden är med paketet subfiles. Denna är generellt sett att föredra om den är tillgänglig, och fungerar ungefär som metod ett men med undantaget att man även kan kompilera underfilerna var för sig, om man är intresserad av detta. Det kan vara lämpligt att göra om man bara redigerar ett kapitel, och inte vill kompilera hela projektet under tiden. Paketet subfiles finns på CTAN men inte i Chalmers datorsystem eller i TeX Live.

Rekommenderade paket

Eftersom TeX är Turingkomplett kan i princip allt göras med språket, även om det främst är tänkt för typsättning. Således kan det mesta i typsättningsväg lösas relativt enkelt. Kan det inte det, och problemet man vill lösa är ett relativt vanligt problem, så finns det sannolikt ett paket som löser problemet. Dessa paket finns listade på CTAN (se sida 3) och kan, om man använder TeX Live, installeras med verktyget `tlmgr`.

Paket i kursiv stil finns inte på Chalmers-datorer utan måste installeras manuellt i hemmappen. Dokumentation för alla paket finns på CTAN och kan även visas genom att skriva `texdoc` *`paketnamn`* i terminalen.

Allmänt nyttiga paket

Dessa paket finns i princip alltid och tillhandahåller så grundläggande funktionalitet att man alltid bör använda dem. Det handlar om allt från tecken- och typsnittskodning till avstavning och interna länkar.

babel översätter interna strängar (till exempel ”Referenser” och ”Sammanfattning”) till det språk som önskas och gör det möjligt för dessa språk att avstavas ordentligt. Paketet är ett måste då man använder LaTeX, men har ersatts av polyglossia i XeLaTeX.

fixltx2e korrigerar en del buggar i LaTeX 2_ε-kärnan och gör till exempel matematikkommandon robusta. Inkludera alltid.

[1] http://hginit.com/

fontenc låter användaren välja typsnittskodning. Från början använder LaTeX `OT1`, som inte innehåller icke-anglikanska tecken så som *å*, *ä* och *ö*. Detta medför problem när det gäller bland annat avstavning, och man bör därför istället använda `T1`. Detta diskuteras närmre på sidan 9.

hyperref gör om innehållsförteckningar, referenser och URIer till riktiga länkar i de fall detta stöds (det vill säga om slutformatet är PDF). Dessutom skapar den inbyggda innehållsförteckningar som kan användas för att navigera i dokumentet i en del PDF-läsare (till exempel Skim och Acrobat Reader). Ett oumbärligt paket som alltid bör inkluderas.

inputenc är ett paket som låter användaren berätta för LaTeX vilken teckenkodning indatan sparats med. De vanigaste inställningarna är `utf8` och `latin1`, men många andra teckenkodningar stöds också. Detta paket introducerades lite kort på sidan 9.

lmodern inkluderar typsnittet *Latin Modern* för att ersätta *Computer Modern*, standardtypsnittet i LaTeX. Latin Modern ser i princip likadant ut som Computer Modern, men är skarpare och har fler glyfer. Det här paketet motiveras lite närmre på sidan 25.

nag försöker upptäcka saker som är *bad practice*, saker som kanske inte fungerar som man tänkt sig, och liknande. Bör inkluderas med alternativen `l2tabu` och `orthodox`:

```
\usepackage[l2tabu,orthodox]{nag}
```

Paket för snyggare typsättning

Även om LaTeX är mycket bra på att typsätta så blir det ibland inte så snyggt som man kanske kan önska. Vid dessa tillfällen kan man applicera diverse olika paket för att få hjälp med detta. Det gäller till exempel tabeller, figurtexter, SI-enheter och matematik.

booktabs är ett paket som gör det möjligt att skapa mycket snygga tabeller. Paketet diskuteras redan på sidan 20, och är i princip ett måste om man ska inkludera tabeller i sitt LaTeX-dokument.

caption gör det möjligt att ändra stilen på figur- och tabelltexter så att de syns tydligare och inte smälter in i texten.

siunitx gör det enklare att typsätta SI-enheter, decimaltal och vinklar på ett korrekt sätt för icke engelskspråkiga rapporter. Mycket användbart paket som tyvärr inte finns på Chalmers datorer. Förklaras lite kort på sidan 44.

Paket för matematiktypsättning

Som teknisk matematiker (eller fysiker) kommer stora delar av de rapporter man skriver oundvikligen innehålla matematik. Även om LaTeX för sig självt är relativt bra på att typsätta matematik kan det ibland behövas några tillägg för att göra det enklare. Många av dessa tillhandahålls av $\mathcal{AMS}$ i det som kallas $\mathcal{AMS}$LaTeX, men det finns fler relevanta matematikpaket.

amscd ger möjlighet att skapa kommutationsdiagram med LaTeX:

$$\begin{array}{ccccccc} \operatorname{cov}(\mathcal{L}) & \longrightarrow & \operatorname{non}(\mathcal{K}) & \longrightarrow & \operatorname{cf}(\mathcal{K}) & \longrightarrow & \operatorname{cf}(\mathcal{L}) \\ \downarrow & & \uparrow & & \uparrow & & \downarrow \\ \operatorname{add}(\mathcal{L}) & \longrightarrow & \operatorname{add}(\mathcal{K}) & \longrightarrow & \operatorname{cov}(\mathcal{K}) & \longrightarrow & \operatorname{non}(\mathcal{L}) \end{array}$$

amsmath gås igenom i del III och är i princip oumbärligt om man ska typsätta matematik med LaTeX. Inkludera alltid detta paket om du skriver något som kan tänkas innefatta ekvationer.

amssymb definierar en hel del matematiska symboler, till exempel `\therefore` ($\therefore$) och `\ggg` ($\ggg$), och kommandot `\mathbb` som ger krittavletecken (som används för de grundläggande talmängderna).

amsthm definierar omgivningar för att typsätta teorem, satser, lemman, bevis och dylikt. Användbart i vissa sammanhang, främst för att typsätta föreläsningsanteckningar eller uppgifter.

fixmath korrigerar typsättningen av grekiska bokstäver, så att även dessa typsätts kursivt (som variabler). Gör det även möjligt att typsätta matematik fetstilt, med hjälp av kommandot `\mathbold`.

mathtools lagar några småfel i amsmath och definierar kommandon som till exempel `\DeclarePairedDelimiter`, som introducerades på sidan 37.

Paket för grafik

Grafik i form av figurer eller illustrationer är givetvis en viktig del av många rapporter, vare sig det är figurer av uppställningar, plottar eller flödesscheman. Det finns tre relevanta paket när det gäller grafik i LaTeX; ett som används för att importera grafik från externa filer och två som används för att rita direkt med LaTeX-kommandon.

graphicx förklaras kort i del IV på sidan 45 och gör det möjligt att inkludera figurer från externa filer i sitt LaTeX-dokument. Mycket användbart om man har data från till exempel MATLAB eller Mathematica.

tikz nämns också i del IV och gör det möjligt att rita vektorbaserade figurer direkt i LaTeX. Mycket användbart om man vill rita exempelvis uppställningar, enklare illustrationer eller flödesscheman, men kan användas till otroligt mycket mer. Fungerar endast med pdfLaTeX i PDF-läge eller moderna varianter som XeLaTeX.

pstricks är en slags motsvarighet till tikz som endast fungerar med de LaTeX-varianter som genererar DVI- eller PS-filer. Inte lika användbart som tikz eftersom det inte fungerar med pdfLaTeX i PDF-läge.

Paket för bibliografier

Även om BibTeX ofta är tillräckligt för att kunna typsätta dokument som ska skickas till journaler eller förläggare (eftersom verktyget inkluderar många vanliga engelskspråkiga bibliografistilar) så är det inte alltid tillräckligt. Till exempel så rekommenderar Chalmers bibliotek en bibliografistil som inte finns med i BibTeX och som till råga på allt ska vara på svenska. Sådana problem löses enkelt av diverse paket.

chscite är ett paket som skapats för att implementera de rekommendationer Chalmers Bibliotek publicerat (Chalmers Bibliotek 2010) i BibTeX. Rekommenderas absolut för alla sorters rapporter, artiklar och publikationer du kan tänkas producera under din studietid, så länge det inte finns krav på stil från annat håll. Diskuteras även på sidan 52.

Paket som löser problem

Ibland vill man göra något som är väldigt svårt att göra med vanlig LaTeX-kod, till exempel ändra sidstorlek eller skapa underfigurer. Då många sådana problem är vanligt förekommande finns det ofta paket som löser dem.

enumitem gör det möjligt att definiera egna sorters listor, och modifierar `enumerate` så att man även kan specificera en egen etikett, återuppta numreringen från förra listan, och mycket mer.

float definierar ett kommando `\newfloat` som skapar nya sorters flytande objekt (vilka diskuterats på sidan 18).

geometry kan användas för att ändra en del mått i LaTeX (till exempel sidans storlek eller marginalerna) om så önskas. Oftast behöver man inte göra detta, eftersom dokumentklassen har definierat de mått den har av en anledning. Kan vara värdefullt om man vill skapa dokument för A5-eller A6-papper (eller andra ovanliga storlekar).

multicol löser problemet med att typsätta text i kolumner på ett mycket bättre sätt än standardklasserna, med hjälp av omgivningen `multicols`. Paketet stödjer bland annat korrekt balanserade kolumner.

multirow gör det möjligt att skapa tabellceller som spänner över flera rader.

sidecap definierar omgivningen `SCfigure` som typsätter en figur med figurtexten brevid istället för under eller över figuren. Kan vara användbart om man har smala figurer med lång figurtext.

subfig definierar kommandon som möjliggör skapandet av ”underfigurer”, det vill säga grupper av relaterade figurer som alla kan refereras till antingen som grupp (till exempel ”Figur 1”) eller individuellt (till exempel ”Figur 1a”).

varioref gör det möjligt att, med hjälp av kommandot `\vref`, skapa korsreferenser som inte bara refererar till etiketterna med nummer utan även till den sida figuren eller tabellen finns på. Detta görs på ett intelligent sätt, så att referensen blir ”figur X på nästa sida” om figuren är på nästa sida, och så vidare. Mycket relevant paket, speciellt i större rapporter.

wrapfig skapar ett nytt flytande objekt `wrapfig` som placerar figuren till höger eller vänster på sidan och låter övrig text ”flyta” runt figuren.

xspace definierar ett kommando `\xspace` som kan läggas till på slutet av kommandodefinitioner för att de ska bete sig snällare i brödtext, så att man slipper skriva `{}` efter kommandot.

Paket för att ändra utseende

Standardklasserna lämnar ofta en del att önska i termer av utseende. Vill man göra något åt detta kan man använda paket som utformats för att låta dig ändra stilen av vissa element, till exempel rubriker eller sidhuvuden[1].

fancyhdr låter dig förändra sidhuvud och sidfot för att införa snyggare, mer informativa eller tydligare stilar. Till exempel så kan texten i sidhuvudet ändras för att visa sidnummer och kapitel, medan sidfoten ändras för att visa till exempel kontakt- eller copyrightinformation.

[1] Vill man lösa problemet på riktigt ska man givetvis använda en dokumentklass som ser bättre ut istället.

titlesec gör det möjligt att ändra stilen på rubrikerna i dokumentet, till exempel genom att byta textstil eller sättet rubriken typsätts.

tocloft definierar kommandon för att styra utseendet av innehållsförteckningen.

Andra specialiserade paket

För en del snäva områden så som kvantfysik eller datavetenskap finns det specialiserade paket som löser specifika uppgifter. Dessa kan med fördel användas om man skriver rapporter inom området, eller rör vid ämnet i någon inlämningsuppgift.

algorithmic är ett paket för typsättning av algoritmer och pseudokod.

braket definierar kommandon för att handskas med braket-notationen som används inom kvantfysik. Kommandot `\Braket` kan således resultera i följande ekvation:

$$\left\langle \phi \middle| \frac{\partial^2}{\partial t^2} \middle| \psi \right\rangle$$

Paketet definierar även kommandot `\Set`, för att på liknande sätt typsätta mängddefinitioner:

$$\left\{ x \in \mathbf{R}^2 \;\middle|\; 0 < |x| < 5 \right\}$$

listings kan användas för att inkludera programkod i LaTeX-dokument. Det finns möjlighet att skriva ut radnummer, lägga till ramar, associera figurtexter och även en mycket enkel syntaxfärgning (eng. *syntax highlighting*). Fungerar inte om koden innehåller svenska tecken, även om paketet ***listingsutf8*** försöker lösa detta. Använder man XeLaTeX uppstår inte detta problem.

minted kan sägas vara en förbättring av **listings** som använder Python-programmet *Pygments* för att även sätta färg på koden. Ett bra alternativ om man ska inkludera kod och vill ha fullständig syntaxfärgning. Paketet lider dock av samma problem med svenska tecken som **listings**.

todonotes låter dig infoga (synliga) ”att göra”-anteckningar i ditt dokument, tillsammans med en lista över dessa. Paketet innehåller även ett kommando `\missingfigure`, som kan användas för att markera en saknad figur.

Andra TeX-baserade projekt

LaTeX är inte det enda användbara TeX-baserade projektet, även om det är det makropaket som används mest i praktiken. Det finns alternativ som baseras på LaTeX men löser problem (till exempel XeLaTeX och LuaTeX) men även alternativa makropaket som är mycket olika LaTeX i sin struktur (till exempel ConTeXt). Dessutom finns det projekt för att utveckla LaTeX 2_ε och göra vissa saker som till exempel utveckling av dokumentklasser mycket enklare.

Unicode-baserade XeLaTeX

XeLaTeX är, precis som pdfLaTeX, en TeX-kompilator. Denna kan användas på samma sätt som vanliga TeX och pdfLaTeX (dvs. den kan köras på i princip samma kod) med hjälp av dess program, `xelatex`.

Skillnaden mellan pdfLaTeX och XeLaTeX som gör att man kanske föredrar det senare är relativt stor, sett till de djupa delarna av TeX: XeLaTeX förutsätter, till skillnad från pdfLaTeX, att all indata

är UTF-8 och kan därmed läsa UTF-8-dokument på ett korrekt sätt. Detta innebär bland annat att saker som inte fungerar i pdfLaTeX ens med inputenc, till exempel svenska tecken i kodlistingar eller inkluderade filer, kommer att fungera utmärkt med XeLaTeX.

Dessutom kan man med hjälp av speciella kommandon välja typsnitt på ett mycket enklare sätt, eftersom XeLaTeX stödjer OTF- och TTF-typsnitt, och kan hitta alla typsnitt som är installerade på datorn, inte bara de som finns i form av LaTeX-paket.

En relativt övergripande bild av vad som är nytt i XeLaTeX ges av Robertson (2011); för det mesta kan man förutsätta att LaTeX-kod kommer att fungera precis likadant i XeLaTeX.

Skripta med LuaTeX

LuaTeX är en kombination av programmeringsspråket Lua och typsättningsspråket TeX. Introduktionen till LuaLaTeX, som är LuaTeXs motsvarighet till vanliga LaTeX, beskriver systemet som uppföljaren till pdfLaTeX:

> *It is the designated successor of pdfTeX and includes all of its core features: direct generation of* PDF *files with support for advanced* PDF *features and micro-typographic enhancements to TeX typographic algorithms.*
>
> *(Pégourié-Gonnard 2010)*

I många avseenden är LuaTeX mycket likt XeLaTeX; man kan använda konventionella typsnitt med båda motorerna (med paketet fontspec), och de hanterar båda UTF-8 mycket bättre än sina företrädare. Skillnaden är att LuaTeX även gör det möjligt att bädda in Lua-kod direkt i dokumentet, vilket gör en del saker mycket enklare eftersom man har tillgång till ett fullgott skriptspråk.

Ett alternativ: ConTeXt

ConTeXt är ett alternativt makropaket för TeX som utvecklats parallellt med LaTeX men med en annan inriktning. Medan LaTeX försöker isolera användaren från typografiska beslut, vilket gör det lämpligt för att till exempel skicka in artiklar till förlag, så försöker ConTeXt ge användaren en lite mer strukturerad tillgång till de typografiska funktionerna i TeX.

Det finns en del artiklar som jämför ConTeXt och LaTeX (till exempel Hoekwater 1998), men kontentan av det hela är att LaTeX är vanligare i akademiska kretsar och bättre på att typsätta matematik, och att man därför oftast bör hålla sig till LaTeX. Är man trots detta intresserad av ConTeXt bör man referera till ConTeXt-manualen (Hagen 2001).

Referenser

American Mathematical Society (1999) User's Guide for the amsmath Package.
ftp://ftp.ams.org/ams/doc/amsmath/amsldoc.pdf.

Chalmers Bibliotek (2010) Referensguide.
http://www.lib.chalmers.se/education/guides/references/.

Chicago Editorial Staff (2010) *The Chicago Manual of Style*. 16:e upplagan. Chicago: University of Chicago Press.

Christoffersson, K. et al. (1998) *Språkguiden*. [Elektronisk] Linköping: Linköpings Universitet.
http://www.student.liu.se/service/sprakguide/1.265737/sprakguiden.pdf.

Downes, M. (2002) Short Math Guide for LaTeX.
ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf.

Eijkhout, V. (1992) *TeX by Topic*. [Elektronisk] Reading, Massachusetts: Addison-Wesley.
http://eijkhout.net/texbytopic/texbytopic.html.

Fear, S. (2005) Publication quality tables in LaTeX.
http://mirrors.ctan.org/macros/latex/contrib/booktabs/booktabs.pdf.

Fenn, J. (2006) Managing Citations and Your Bibliography with BibTeX. *The PracTeX Journal*, nr 4.
http://www.tug.org/pracjourn/2006-4/fenn/fenn.pdf.

Fenn, J. och Trettin, M. (2007) An essential guide to LaTeX 2_ε usage.
http://mirrors.ctan.org/info/l2tabu/english/l2tabuen.pdf.

Flynn, P. (2006) Rolling your own Document Class: Using LaTeX to keep away from the Dark Side. *The PracTeX Journal*, nr 4.
http://tug.org/pracjourn/2006-4/flynn/flynn.pdf.

Grätzer, G. A. (1996) *Math into LaTeX*. [Elektronisk] Boston: Birkhäuser.
http://mirrors.ctan.org/info/mil/mil.pdf.

Hagen, H. (2001) ConTeXt the manual.
http://www.pragma-ade.com/general/manuals/cont-eni.pdf.

Hoekwater, T. (1998) Comparing ConTeXt and LaTeX. *MAPS Journal*.
http://maps.aanhet.net/maps/pdf/20_42.pdf.

Knuth, D. (1986) *The TeXbook*. Reading, Massachusetts: Addison-Wesley.

Lamport, L. (1994) *LaTeX: A Document Preparation System*. 2:a upplagan. Reading, Massachusetts: Addison-Wesley.

Mertz, A. och Slough, W. (2007) Graphics with Ti*k*Z. *The PracTeX Journal*, nr 1.
http://www.tug.org/pracjourn/2007-1/mertz/mertz.pdf.

Mittelbach, F. et al. (2004) *The LaTeX Companion*. 2:a upplagan. Reading, Massachusetts: Addison-Wesley.

Oetiker, T. et al. (2011) The Not So Short Introduction to LaTeX 2ε.
http://mirrors.ctan.org/tex-archive/info/lshort/english/lshort.pdf.

Pakin, S. (2009) The Comprehensive LaTeX Symbol List.
http://mirrors.ctan.org/info/symbols/comprehensive/symbols-a4.pdf.

Patashnik, O. (1988a) Designing BibTeX Styles.
http://mirrors.ctan.org/biblio/bibtex/contrib/doc/btxhak.pdf.

Patashnik, O. (1988b) BibTeXing.
http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf.

Pégourié-Gonnard, M. (2010) A guide to LuaLaTeX.
http://mirrors.ctan.org/info/luatex/lualatex-doc/lualatex-doc.pdf.

Robertson, W. (2006) Productivity with macros and packages. *The PracTeX Journal*, nr 3.
http://tug.org/pracjourn/2006-3/robertson/robertson.pdf.

Robertson, W. (2011) The XeTeX reference guide.
http://mirrors.ctan.org/info/xetexref/XeTeX-reference.pdf.

Språkrådet (2008) *Svenska skrivregler*. 3:e upplagan. Solna: Liber AB.

Tantau, T. (2010) The Ti*k*Z and PGF Packages.
http://mirrors.ctan.org/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf.

The LaTeX3 Projext (1999) LaTeX 2ε for class and package writers.
http://www.latex-project.org/guides/clsguide.pdf.

Thành, H. T. (2000) *Micro-typographic extensions to the TeX typesetting system*. [Elektronisk] Brno: Masaryk University. (Doktorsavhandling).
http://www.pragma-ade.com/pdftex/thesis.pdf.

von Schultz, C. (2005) Rekommendationer för LaTeX-dokument.
http://web.student.chalmers.se/~von/latex/rekommendationer.pdf.

Voß, H. (2010) Math mode.
http://mirrors.ctan.org/info/math/voss/mathmode/Mathmode.pdf.

Williams, P. och Schnier, T. (2008) The Harvard Family of Bibliography Styles.
http://mirrors.ctan.org/macros/latex/contrib/harvard/harvard.pdf.

Wright, J. (2011) siunitx — A comprehensive (SI) units package.
http://mirrors.ctan.org/macros/latex2e/exptl/siunitx/siunitx.pdf.

A
En enkel mall

Följande mall är tänkt att användas för att skriva enklare rapporter på Chalmers Tekniska Högskola. Eftersom LaTeX-installationen på deras datorer saknar en del paket (siunitx och chscite) är dessa bortkommenterade. Vill man ändå använda dem kan man installera dessa i sin hemmapp, under `~/texmf/`[1].

Kommentarerna och texten i mallen bör förklara och motivera vad som görs vilka paket som inkluderas någorlunda. Om något är oklart, referera till motsvarande del av den här introduktionen. Hela mallen finns dessutom tillgänglig som en `.tex`-fil för nedladdning[2].

```
\documentclass[a4paper,11pt]{article}
% Allmängiltiga paket
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{lmodern}
\usepackage[swedish]{babel}
\usepackage{microtype}
\usepackage{icomma}
\usepackage{graphicx}
\usepackage{fixltx2e}
%\usepackage[l2tabu]{nag} % finns ej på Chalmers-datorer
% Matematikpaket
\usepackage[intlimits]{amsmath}
\usepackage{amssymb}
\usepackage{mathtools}
%\usepackage{siunitx} % finns ej på Chalmers-datorer
\usepackage{textcomp}
% Snyggare tabeller
\usepackage{booktabs}
% Snyggare figurtexter
\usepackage[font=small,%
            format=plain,%
            labelfont=bf,%
            textfont=it]{caption}
```

[1] Detta förklaras närmre av Oetiker et al. (2011, ss. 89–90)

[2] http://blog.sigurdhsson.org/projects/latexbok.html

```
% Länkade och intelligenta referenser
\usepackage{hyperref}
\usepackage[swedish]{varioref}
% Chalmers-källförteckning
%\usepackage{chscite} % finns ej på Chalmers-datorer

% Inställningar för siunitx
%\sisetup{locale=DE}

% Kommandodefinitioner
% \abs{}, \norm{} – absolutbelopp och norm
% \frac omdefinierad för att se bättre ut
% \d{} – integraldifferentialen
\makeatletter
\DeclarePairedDelimiter\abs{\lvert}{\rvert}
\DeclarePairedDelimiter\norm{\lVert}{\rVert}
\renewcommand{\frac}[2]{\genfrac{}{}{}{}%
            {\displaystyle #1}{\displaystyle #2}}
\renewcommand\d[1]{\ensuremath{\;\mathrm{d}#1%
                  \@ifnextchar\d{\!}{}}}
\makeatother

% Dokumentets metadata
\title{Ett exempeldokument}
\author{Simon Sigurdhsson}
\date{9 augusti 2011}

% Här börjar dokumentet
\begin{document}
    \maketitle
    \begin{abstract}
        Man bör alltid ha en kort sammanfattning (eng.
        \emph{abstract}) till sina rapporter och
        artiklar. En sådan skapas med omgivningen
        \texttt{abstract}.
    \end{abstract}

    % En innehållsförteckning vill man ha ibland:
    \tableofcontents
    \newpage

    \section*{Introduktion}
    Det här avsnittet är tänkt att innehålla en kort
    introduktion till rapporten som förklarar dess syfte
    och mål. Man behöver så klart inte alltid en
    introduktion, men om man har en så bör den vara
    onumrerad, och ska således skapas med
    \texttt{\textbackslash section*}.
```

```
    \section{Första avsnittet}
    Som du ser kan man radbryta sin text i \LaTeX{} utan
    att det påverkar resultatet. Gör det --- det blir mer
    lättläst. Vissa textredigerare kan göra detta
    automatiskt.

    Det fungerar i ekvationer också:
    \begin{equation}
        f(x) = \sin x
        \implies
        F(x) = \int\! f(x)\d{x} = \cos x + c.
    \end{equation}

    \subsection{Ett underavsnitt}
    Man kan även infoga tabeller i \LaTeX{} (se tabell
    \ref{tab}, till exempel) och figurer (figur
    \ref{fig}). Dessa kan man referera till på olika
    sätt, till exempel med kommandot
    \texttt{\textbackslash vref}: ``figur \vref{fig}''.

    \begin{table}[tp]
        \centering
        \label{tab}
        \caption{En tabell över fiktiva priser}
        \begin{tabular}{lp{0.4\textwidth}r}
            \toprule
            Namn & Beskrivning & Pris (\$) \\
            \midrule
            Cykel & Enkelt tvåhjuligt fordon utan motor
            för transport av enstaka personer över korta
            sträckor & 200 \\
            Polkagris & Röd-vit-randig sötsak med mycket
            hård konsistens, tillverkad i Gränna & 2,50\\
            \bottomrule
        \end{tabular}
    \end{table}

    \begin{figure}[bp]
        \centering
        \includegraphics[width=0.8\textwidth%
                         ]{bilder/rulltarta.png}
        \caption{En mycket god rulltårta}
        \label{fig}
    \end{figure}
\end{document}
```